11:13:37 OCA PAD AMENDMENT - PROJECT HEADER INFORMATION 09/18/96

Active

Project #: C-36-690 Cost share #: Rev #: 3
Center # : 10/24-6-R7554-0A0 Center shr #: OCA file #:
Work type : RES
Contract#: CCR-9200878 Mod #: AMENDMENT #002 Document : GRANT
Prime #: Contract entity: GTRC

Subprojects ? : N CFDA: 47.070
Main project #: PE #: N/A

Project unit: COMPUTING Unit code: 02.010.300
Project director(s):
VENKATESWARAN H COMPUTING (404)894-2589

Sponsor/division names: NATL SCIENCE FOUNDATION / GENERAL
Sponsor/division codes: 107 / 000

Award period: 920715 to 970131 (performance) 970430 (reports)

Sponsor amount	New this change	Total to date
Contract value	0.00	69,951.00
Funded	0.00	69,951.00
Cost sharing amount		0.00

Does subcontracting plan apply ?: N

Title: STUDIES IN COMPLEXITY THEORY: A CIRCUIT BASED APPROACH

PROJECT ADMINISTRATION DATA

OCA contact: Michelle A. Starmack 894-4820

Sponsor technical contact	Sponsor issuing office
YECHEZKEL ZALCSTEIN	GLORIA YOUNG
(703)306-1914	(703)306-1212
NATIONAL SCIENCE FOUNDATION	NATIONAL SCIENCE FOUNDATION
4201 WILSON BOULEVARD	4201 WILSON BOULEVARD
ARLINGTON, VA 22230	ARLINGTON, VA 22230

Security class (U,C,S,TS) : U ONR resident rep. is ACO (Y/N): N
Defense priority rating : N/A NSF supplemental sheet
Equipment title vests with: Sponsor GIT X

Administrative comments -
AMENDMENT NO. 002 EXTENDS EXPIRATION DATE OF GRANT TO 1/31/97. THE FINAL REPORT IS NOW DUE 4/30/97

(2)

Closeout Notice Date 02-MAY-1997

Project Number C-36-690 Doch Id 44794

Center Number 10/24-6-R7554-0A0

Project Director VENKATESWARAN, H.

Project Unit COMPUTING

Sponsor NATL SCIENCE FOUNDATION/GENERAL

Division Id 3393

Contract Number CCR-9200878 Contract Entity GTRC

Prime Contract Number

Title STUDIES IN COMPLEXITY THEORY: A CIRCUIT BASED APPROACH

Effective Completion Date 31-JAN-1997 (Performance) 30-APR-1997 (Reports)

Closeout Action:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	N	
Final Report of Inventions and/or Subcontracts	N	
Government Property Inventory and Related Certificate	N	
Classified Material Certificate	N	
Release and Assignment	N	
Other	N	

Comments
LETTER OF CREDIT APPLIES. 98A SATISFIES PATENT REPORT.

Distribution Required:

Project Director/Principal Investigator	Y
Research Administrative Network	Y
Accounting	Y
Research Security Department	N
Reports Coordinator	Y
Research Property Team	Y
Supply Services Department	Y
Georgia Tech Research Corporation	Y
Project File	Y

To NSF Program: _____

APPENDIX VIII

Annual NSF Grant Progress Report

PI Name: Dr. H. Venkateswaran

NSF Award Number: CCR 9200878

PI Institution: Georgia Institute of Technology

PI Address: Georgia Institute of Technology
College of Computing
Atlanta, GA. 30332-0280

Date: 4-30-93

I certify that to the best of my knowledge (1) the statements herein (excluding scientific hypotheses and scientific opinions) are true and complete, and (2) the text and graphics in this report as well as any accompanying publications or other documents, unless otherwise indicated, are the original work of the signatories or individuals working under their supervision. I understand that the willful provision of false information or concealing a material fact in this report or any other communication submitted to NSF is a criminal offense (U.S.Code, Title 18, Section 1001.)

Signature: _____

Please include the following information:

1. A brief summary of overall progress, including results obtained to date, their relationship to the general goals of the award and their significance to science;
2. an indication of any current problems or favorable or unusual developments;
3. a brief summary of work to be performed during the next year of support if changed from the original proposal; and
4. any other information pertinent to the type of project supported by NSF or as specified by the terms and conditions of the grant, including a statement describing the contribution of the research in the area of education and human resources development.

If applicable, please attach a copy of any updated human subject or animal subject certification.
[Attach additonal sheets as necessary.]

Annual Project Report for NSF Grant CCR-9200878

TITLE: Studies in Complexity Theory: A Circuit Based Approach

PI: H. Venkateswaran

This NSF grant was awarded effective from July 15, 1992. It mainly provides support for a research assistant and travel to conferences. A Ph.D. student Ms. Rimli Sengupta who is doing her dissertation research in the area of Computational Complexity theory is currently supported by this grant.

This grant supports research work that studies several interesting issues in complexity theory using circuits as the unifying framework. The main focus of our work is the understanding of the relationships among complexity classes. In the circuit setting, this would involve proving lower bounds on circuit resources such as size and depth. Because of the difficulty in proving such lower bounds for general Boolean circuits, one considers restricted circuits (as, for example, monotone Boolean circuits) with the hope that techniques to prove bounds in the restricted setting can be adapted to the general setting. We are taking two different approaches to studying restricted circuits. One approach considers axiomatic restrictions of a Boolean algebra with the objective of identifying and isolating the computational power lent by the axioms such as idempotence and absorption. A specific function, namely the Boolean permanent is used for these studies. We have been able to prove exponential size lower bounds for computing the Boolean permanent in two such restricted models. This approach is motivated by a similar one taken by Jerrum and Snir (Some exact complexity results for straight-line computations over semi-rings, JACM, V. 29, 1982, pp. 874-897). They develop a combinatorial technique to obtain an exponential size lower bound for computing the permanent using only semi-ring axioms. Our lower bounds improve this to show that the exponential lower bound also holds in the presence of some additional axioms of Boolean algebra. We are currently looking into some specific semi-rings such as those defined with the operations (MAX, CONCAT) and the operations $(+, \times)$. By defining complexity classes over such semi-rings, we expect to understand the relationships between them and the Boolean classes.

Another approach that we are taking is to consider restrictions in the circuit model itself. Monotonicity, for example, is such a restriction in which complemented inputs are not supplied. An interesting restriction is to specify that the parse-graphs of circuits are trees. (A parse-graph of a circuit is a subgraph of the circuit that corresponds to a monomial of the formal polynomial associated with the circuit.) We show that circuits for natural problems such as Warshall circuits for Transitive closure, Cocke-Kasami-Younger circuits for Context-free language recognition, and

the NC^1 circuits for Parity have this property. It is interesting to note that the standard polynomial size Boolean circuit for perfect matching in bipartite graphs does not seem to obey this restriction. An interesting, but difficult question here is to show a super polynomial size lower bound for the permanent problem in this restricted model. Another interesting restriction is to impose parsimony on Boolean circuits. This restriction seems to capture some connections between Boolean complexity classes and counting complexity classes. Define a Boolean circuit computing a Boolean function f to be *parsimonious* if an arithmetization of the circuit computes $\#f$, the number of solutions of f. The existence of polynomial size parsimonious Boolean circuits for the bipartite perfect matching problem would have unexpected consequences such as $\#P = FP$, where FP is the function class corresponding to the class P. This shows that it is unlikely that polynomial size Boolean circuits for the bipartite perfect matching problem are parsimonious. We are also studying Boolean circuits that are restricted to use only $(EXCLUSIVE\ OR, AND)$ gates and which are not provided with complemented inputs or the constant 1. It can be shown that such circuits of depth 2 to compute the OR function requires exponential size. Currently, we are looking at the complexity classes defined by families of such restricted circuits and their relationships to other classes.

Education and Human Resources: The circuit approach is very appealing because it gives a simple and unifying view of many of the interesting results of complexity theory. It also has led to the development of new proof techniques for addressing some of the more difficult questions in this area. I am currently teaching a graduate seminar course that covers some of these important results in Complexity Theory using the circuit framework. This grant supports a woman PhD student as a research assistant. She is currently working on problems that fit into the goals of this award. I expect her to be able to complete a dissertation in complexity theory in the coming year.

Travel: The grant supported the travel of the PI and the student to attend the thirty-second Symposium on Foundations of Computer Science (FOCS) held in Pittsburgh in October 1992.

C-36-690
N/a

Annual Project Report for NSF Grant CCR-9200878

TITLE: Studies in Complexity Theory: A Circuit Based Approach

PI: H. Venkateswaran

Introduction: This NSF grant supports research work in computational complexity theory using circuits as the unifying framework. Some of the important complexity questions using the circuit model involve proving lower bounds on circuit resources such as size and depth. Because of the difficulty in proving such lower bounds for general Boolean circuits, one considers restricted circuits (as, for example, monotone Boolean circuits) with the hope that techniques to prove bounds in the restricted setting can be adapted to the general setting. A further contribution of such an investigation would be to quantify the power of properties in the Boolean domain such as negation and multiplicative idempotence in reducing the complexity of computing natural functions.

Technical Contributions: The P.I. in collaboration with a Ph.D. student studied the power of restricted models of Boolean circuits as part of this project. The two important restrictions studied are: withholding multiplicative idempotence and not allowing cancelation.

- It is shown that monotone circuits that do not use the multiplicative idempotence property require exponential size to compute the Connectivity function. Since connectivity can be computed by polynomial size monotone Boolean circuits, this shows that multiplicative idempotence, in the context of monotone circuits, is exponentially powerful. Generalizing a lower bound of Jerrum and Snir (JACM, vol. 29, 1982, pp. 874-897), it was shown that monotone arithmetic circuits that also use multiplicative idempotence to compute the 0-1 permanent function requires exponential size.

- A notion of non-cancellative circuits was introduced that generalizes monotonicity. Using this non-cancellative complexity classes and relationships among them were studied. It was also shown that cancelation does help in saving the complexity of computing certain non-monotone functions.

Education and Human Resources: The circuit approach is very appealing because it gives a simple and unifying view of many of the interesting results of complexity theory. It also has led to the development of new proof techniques for addressing some of the more difficult questions in this area. The following lists the important activities pertaining to educational impact and human resources development directly related to this grant:

- This grant supported a woman PhD student as a research assistant. She worked on research problems that fit into the goals of this award. She completed her dissertation under the supervision of the P.I. and graduated in Summer 1995. She is currently employed as an Assistant Professor at the Rose-Hullman Institute of Technology in Indiana. This student has also given several talks on her work at many places where she interviewed for a job.

- The P.I. has taught a graduate seminar course that covers some of these important results in Complexity Theory using the circuit framework. One of the students who took that seminar course is continuing his Ph.D. in theoretical Computer Science at Cornell. Another student is also working in theoretical Computer Science for his Ph.D. here at Georgia Institute of Technology.

- The P.I. delivered an invited talk at the workshop on Algebraic Methods in Complexity Theory held in Madras, India in December 1994. The title of the talk was, "Boolean Circuits and Complexity Theory". The P.I. plans to write a survey article based on this talk.

Travel: The grant supported the travel of the PI and the student to attend the thirty-second Symposium on Foundations of Computer Science (FOCS) held in Pittsburgh in October 1992. It supported the travel of the student working on this project to attend the STOC and Structures conference held in San Diego, California in May 1993. It also supported the travel of the PI to travel to Madras, India to present an invited talk entitled "Boolean Circuits and Complexity Theory" at a workshop on "Algebraic Methods in Complexity Theory" in December 1994.

Justification for Extension: This NSF grant was awarded effective from July 15, 1992. It provided support for a research assistant and travel to conferences. A Ph.D. student Ms. Rimli Sengupta who did her dissertation research in the area of Computational Complexity theory was supported by this grant. She graduated in Summer 1995. The P.I. seeks a no-cost extension for a further period of six months to bring this work to a logical conclusion. This would involve finishing up the papers that describe the results obtained as part of this research. During this period, it is also proposed to prepare a survey on the topics covered by this work. It may be relevant to note that the P.I. was on leave from Georgia Tech starting January 1995 until March 1996 thus necessitating this request for extension.

Multilinearity Can Be Exponentially Expensive *

Rimli Sengupta and H. Venkateswaran

College of Computing

Georgia Institute of Technology

Atlanta, GA 30332-0280

e-mail : {rimli,venkat}@cc.gatech.edu

March 8, 1996

*This work was supported by NSF grant CCR-9200878.

Abstract

We define a Boolean circuit to be *multilinear* if the formal polynomial associated with it is multilinear. We consider the problem of computing the connectivity function using circuits that are monotone and multilinear. Our main result is that monotone multilinear circuits for this function require exponential size. Since connectivity can be computed by monotone Boolean circuits within size $O(n^3)$, the lower bound establishes that the multilinearity restriction can be exponentially expensive. Moreover, based on the observation that connectivity can be computed by monotone circuits of polynomial size and polynomial degree, the lower bound exhibits an exponential gap between multilinearity and polynomial degree.

A Lower Bound for Boolean Permanent in Bijective Boolean Circuits and its Consequences *

Rimli Sengupta and H. Venkateswaran

College of Computing

Georgia Institute of Technology

Atlanta, GA 30332-0280

e-mail : {rimli,venkat}@cc.gatech.edu

March 8, 1996

*This work was supported by NSF grant CCR-9200878.

Abstract

We identify a new and non-trivial restriction on Boolean circuits called bijectivity and prove an exponential size lower bound for computing the Boolean permanent function in this model. As consequences of this lower bound, we show exponential size lower bounds for: (a) computing the Boolean permanent using monotone multilinear circuits; (b) computing the 0-1 permanent function using monotone arithmetic circuits; and (c) computing the lexicographically first bipartite perfect matching function using circuits over $(min, concat)$. The lower bound arguments for the Boolean permanent function are adapted to prove an exponential lower bound for computing the Hamiltonian cycle function using bijective circuits. We identify a class of monotone functions such that if the counting version of such a function is $\sharp\mathcal{P}$-hard, then there is no polynomial size bijective circuit for the function unless $\mathcal{PH}$ collapses.

Keywords restricted Boolean circuits, bijectivity, size lower bounds, complexity classes, arithmetic circuits, Boolean permanent.

AMS subject classifications 68Q05, 68Q15, 68Q40.

Non-cancellative Boolean Circuits: A Generalization of Monotonicity*

Rimli Sengupta†
Dept. of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803-3999
e-mail: rimli@cs.rose-hulman.edu

H. Venkateswaran
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
e-mail : venkat@cc.gatech.edu

March 8, 1996

Abstract

Cancellations are known to be helpful in efficient algebraic computation of polynomials over fields. We define a notion an cancellation in Boolean circuits and define Boolean circuits that do not use cancellation to be *non-cancellative*. Non-cancellative Boolean circuits are a natural generalization of monotone Boolean circuits. We show that in the absence of cancellation, Boolean circuits require super-polynomial size to compute the determinant interpreted over $GF(2)$. This non-monotone Boolean function is known to be in $\mathcal{P}$. In the spirit of monotone complexity classes, we define complexity classes based on non-cancellative Boolean circuits. We show that when the Boolean circuit model is restricted by withholding cancellation, $\mathcal{P}$ and popular classes within $\mathcal{P}$ are restricted as well, but classes $\mathcal{NP}$ and above remain unchanged.

1 Introduction

Using the power of cancellation to compute more efficiently has been a recurrent theme in the study of computational complexity. Strassen [21] made elegant usage of cancellation to obtain a surprising $O(n^{2.81})$ algorithm for matrix multiplication, improving the obvious $O(n^3)$ algorithm. Valiant [25] showed that usage of cancellations can lead to an exponential gain in size for counting the number of perfect matchings in triangular grid graphs. Nisan [16] showed that if cancellation is inhibited by withholding commutativity, then computing the determinant is as hard as computing the permanent. All of these results are for algebraic computations over the field of rationals. In this paper, we study the power of cancellation in Boolean computation. We define a Boolean circuit to be *non-cancellative* if the formal polynomial associated with the circuit does not have a monomial that has both a literal and its complement. In a non-cancellative circuit, all intermediate computation has influence on the output. In this sense, such circuits cannot perform cancellation. Monotone Boolean circuits, studied extensively in the past [1, 8, 9, 13, 17, 18, 22], have this property. But unlike monotone Boolean circuits, non-cancellative circuits can compute all Boolean functions because the circuit based on the representation of a Boolean function as the disjunction of its prime implicants is non-cancellative.In considering non-cancellative Boolean circuits, we generalize monotonicity such that all Boolean functions are computable and show that lower bounds for functions in the monotone model lead to similar lower bounds for a large class of non-monotone functions in the non-cancellative setting.

*This work was supported by NSF grant CCR-9200878.

†This work was done while the author was at the College of Computing, Georgia Tech, Atlanta GA 30332-0280.

Cancellation Is Exponentially Powerful for Computing the Determinant

Rimli Sengupta*
Dept. of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
e-mail : sengupta@cs.rose-hulman.edu

March 8, 1996

Abstract

Cancellations are known to be helpful in efficient computation of polynomials with positive coefficients. We study the power of cancellation for computing the determinant. For arithmetic circuits over the reals our result is that in the absence of cancellation, computing the determinant is as hard as computing the permanent and that both functions require exponential circuit size. We show a similar result for arithmetic circuits over $GF(2)$.

1 Introduction

Using the power of cancellation to compute efficiently has been a recurrent theme in the study of computational complexity. Strassen [5] made elegant usage of cancellation to obtain a surprising $O(n^{2.81})$ algorithm for matrix multiplication, improving the obvious $O(n^3)$ algorithm. Valiant [7] showed that judicious usage of cancellations can lead to an exponential gain in the size of algebraic circuits that count the number of perfect matchings in triangular grid graphs. These results are about using cancellations to efficiently compute polynomials with positive coefficients (monotone polynomials). For computing monotone polynomials, studying the power of cancellation is equivalent to studying the power of negation since negations can only be used for cancellation. The situation is quite different in the case of polynomials that may have negative coefficients (non-monotone polynomials). There is an important distinction between negation and cancellation for computing non-monotone polynomials: negations are essential but cancellations are not. In this note we show that, as in the case of monotone polynomials, cancellations also help in computing non-monotone polynomials efficiently. The specific non-monotone polynomial we consider is the determinant.

We investigate the power of cancellation over the field of reals and over $GF(2)$. For circuits over both of these algebras we identify a notion of cancellation based on additive inverse. We show that in the absence of cancellation, computing the determinant using circuits over reals is as hard as computing the permanent and that both require exponential size. Since the determinant function is known to be computable within size $O(n^{3.5})$ using circuits over reals [1], our lower bound for the determinant implies that cancellations can be exponentially powerful even for computing non-monotone polynomials. This result also provides an important insight about efficient computation

*This work was done while the author was at the College of Computing, Georgia Tech, Atlanta GA 30332-0280. This work was supported by NSF grant CCR-9200878.

SUMMARY
PROPOSAL BUDGET

ORGANIZATION Georgia Institute of Technology/GTRC				FOR NSF USE ONLY PROPOSAL NO.	DURATION (MONTHS) PROPOSED	 GRANTED
PRINCIPAL INVESTIGATOR/PROJECT DIRECTOR Dr. H. Venkateswaran				AWARD NO.		
A. SENIOR PERSONNEL: PI/PD. Co-PI's, Faculty and Other Senior Associates (List each seperately with title, A.7. show number in brackets)	NSF Funded Person-mos. CAL	 ACAD	 SUMR	Funds Requested By Proposor		Funds Granted By NSF (If Different)
1.						
2.						
3.						
4.						
5.						
6. ( ) OTHERS (LIST INDIVIDUALLY ON BUDGET EXPANATION PAGE)						
7. ( ) TOTAL SENIOR PERSONNEL (1-6)						
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( ) POST DOCTORAL ASSOCIATES						
2. ( ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)						
3. ( ) GRADUATE STUDENTS						
4. ( ) UNDERGRADUATE STUDENTS						
5. ( ) SECRETARIAL CLERCIAL - (IF CHARGED DIRECTLY)						
6. ( ) OTHER						
TOTAL SALARIES AND WAGES (A+B)						
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COST)						
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A+B+C)						
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.)						
TOTAL EQUIPMENT						
E. TRAVEL 1. DOMESTIC (INCL. CANADA AND U.S. POSSESSIONS)				1,500		
2. FOREIGN						
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS $ _____						
2. TRAVEL _____						
3. SUBSISTENCE _____						
4. OTHER						
( ) TOTAL PARTICIPANT COSTS						
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES				1,036		
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION						
3. CONSULTANT SERVICES						
4. COMPUTER SERVICES						
5. SUBAWARDS						
6. OTHER						
TOTAL OTHER DIRECT COSTS						
H. TOTAL DIRECT COSTS (A THROUGH G)				2,536		
I. INDIRECT COSTS (SPECIFY RATE AND BASE)						
TOTAL INDIRECT COSTS 43%				1,090		
J. TOTAL DIRECT AND INDIRECT COSTS (H+I)				3,626		
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.)						
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)						

M. COST-SHARING: PROPOSED LEVEL $	AGREED LEVEL IF DIFFERENT $			
PI/PD TYPED NAME & SIGNATURE* Dr. H. Venkateswaran	DATE 3/12/96	FOR NSF USE ONLY INDIRECT COST RATE VERIFICATION		
ORG. REP. TYPED NAME & SIGNATURE* Michelle Starmack	DATE	Date Checked	Date of Rate Sheet	Initials-DGC

NSF Form 1030 (7/95) Supersedes All Previous Editions

*Signatures required only for revised budget (GPG III.B)

C-36-690 #2

OMB Number 345-0058

NATIONAL SCIENCE FOUNDATION
4201 Wilson Blvd.,
Arlington, VA 22230

BULK RATE
POSTAGE & FEES PAID
National Science Foundation
Permit No. G-69

PI/PD Name and Address

H Venkateswaran
Information and Computer Science
[redacted]
Atlanta, GA 30332-0280
Atlanta GA 30332

NATIONAL SCIENCE FOUNDATION
FINAL PROJECT REPORT

PART I - PROJECT IDENTIFICATION INFORMATION

1. Program Official/Org. Yechezkel Zalcstein - CCR

2. Program Name COMPUTER & COMPUTATION THEORY PROGRAM

3. Award Dates (MM/YY) From: 07/92 To: 01/97

4. Institution and Address
[redacted]
Administration Building
Atlanta GA 30332

5. Award Number 9200878

6. Project Title
Studies in Complexity Theory: A Circuit Based Approach

You are encouraged to submit your Final Project Report electronically through the NSF FastLane home page (www.fastlane.nsf.gov).

This Packet Contains
NSF Form 98A
And 1 Return Envelope

Grant Conditions (Article 17, GC-1, and Article 9, FDP-11) require submission of a Final Project ort (NSF Form 98A) to the NSF program officer no later than 90 days alter the expiration of the rd. Final Project Reports for expired awards must be received before new awards can be made F Grants Policy Manual Section 677).

v, or on a separate page attached to this form, provide a summary of the completed projects and technical information. Be o include your name and award number on each separate page. See below for more instructions.

PART II - SUMMARY OF COMPLETED PROJECT (for public use)

summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the ct title, it should begin with a topic sentence stating the project's major thesis. The summary should include, if pertinent project being described, the following items:

- primary objectives and scope of the project
- techniques or approaches used only to the degree necessary for comprehension
- findings and implications stated as concisely and informatively as possible

Please See Attached Sheets

PART III - TECHNICAL INFORMATION (for program management use)

eferences to publications resulting from this award and briefly describe primary data, samples, physical collections, tions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available research community. Provide the NSF Invention Disclosure number for any invention.

copies of papers attached

y to the best of my knowledge (1) the statements herein (excluding scientific hypotheses and scientific opinion) are true and complete, and text and graphics in this report as well as any accompanying publications or other documents, unless otherwise indicated, are the original of the signatories or of individuals working under their supervision. I understand that willfully making a false statement or concealing a ial fact in this report or any other communication submitted to NSF is a criminal offense (U.S. Code, Title 18, Section 1001).

	April 28, 1997
Principal Investigator/Project Director Signature	**Date**

IMPORTANT:
MAILING INSTRUCTIONS
Return this *entire* packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the *Attention block* on the envelope.

PART IV -- FINAL PROJECT REPORT -- SUMMARY DATA ON PROJECT PERSONNEL

(To be submitted to cognizant Program Officer upon completion of project)

The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in resonse to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the nformation.

Please enter the numbers of individuals supported under this grant.
Do not enter information for individuals working less than 40 hours in any calendar year.

	Senior Staff		Post-Doctorals		Graduate Students		Under-Graduates		Other Participants[1]	
	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.
A. Total, U.S. Citizens										
B. Total, Permanent Residents	1									
U.S. Citizens or Permanent Residents[2]:										
American Indian or Alaskan Native										
Asian	1									
Black, Not of Hispanic Origin										
Hispanic										
Pacific Islander										
White, Not of Hispanic Origin										
C. Total, Other Non-U.S. Citizens						1				
Specify Country 1. India						1				
2.										
3.										
D. Total, All participants (A + B + C)	1					1				
Disabled[3]										

☐ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

[1] Category includes, for example, college and precollege teachers, conference and workshop participants.

[2] Use the category that best describes the ethnic/racial status fo all U.S. Citizens and Non-citizens with Permanent Residency. *(If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)*

[3] A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. *(Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens.")*

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America and who maintains cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia or the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific territories of Guam, American Samoa, and the Northern Marinas; the U.S. Trust Territory of Palau; the islands of Micronesia and Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

Summary of Completed Project for NSF Grant CCR-9200878

TITLE: Studies in Complexity Theory: A Circuit Based Approach

PI: H. Venkateswaran

Introduction: This NSF grant supported research work in computational complexity theory using circuits as the unifying framework. Some of the important complexity questions using the circuit model involve proving lower bounds on circuit resources such as size and depth. Because of the difficulty in proving such lower bounds for general Boolean circuits, this work considered less general circuits. One of the goals was to quantify the power of properties in the Boolean domain such as negation and multiplicative idempotence in reducing the complexity of computing natural functions.

Technical Contributions: The P.I. in collaboration with a Ph.D. student studied the power of restricted models of Boolean circuits as part of this project. The two important restrictions studied are: withholding multiplicative idempotence and not allowing cancelation.

- It was shown that monotone circuits that do not use the multiplicative idempotence property require exponential size to compute the Connectivity function. Since connectivity can be computed by polynomial size monotone Boolean circuits, this shows that multiplicative idempotence, in the context of monotone circuits, is exponentially powerful.

- Generalizing a well-known lower bound of Jerrum and Snir (JACM, vol. 29, 1982, pp. 874-897), it was shown that monotone arithmetic circuits that also use multiplicative idempotence to compute the 0-1 permanent function requires exponential size.

- A notion of non-cancellative circuits was introduced that generalizes monotonicity. Using this non-cancellative complexity classes and relationships among them were studied. It was also shown that cancelation does help in saving the complexity of computing certain non-monotone functions.

- It was shown that in the absence of cancelation, computing the determinant requires exponential size. Previous work had shown that negations are exponentially powerful for computing certain monotone functions. By separating cancelation from negation, this work presented the first example of non-monotone function, namely the determinant, for which cancelations are exponentially powerful.

- A result by Prof. Avi Wigderson showing that $NL \subseteq \oplus L/\text{poly}$ was generalized to the circuit setting which among other things showed that such an inclusion holds for other classes such as SAC^1.

Education and Human Resources: The circuit approach is very appealing because it gives a simple and unifying view of many of the interesting results of complexity theory. It also has led to the development of new proof techniques for addressing some of the more difficult questions in this area. The following lists the important activities pertaining to educational impact and human resources development directly related to this grant:

- This grant supported a woman PhD student as a research assistant. She worked on research problems that fit into the goals of this award. She completed her dissertation under the supervision of the P.I. and graduated in Summer 1995. She is currently employed as an Assistant Professor at the Rose-Hullman Institute of Technology in Indiana. This student has also given several talks on her work at many places where she interviewed for a job.

- The P.I. has taught a graduate seminar course that covers some of these important results in Complexity Theory using the circuit framework.

- The P.I. delivered an invited talk at the workshop on Algebraic Methods in Complexity Theory held in Madras, India in December 1994. The title of the talk was, "Boolean Circuits and Complexity Theory". The P.I. plans to write a survey article based on this talk.

Non-cancellative Boolean Circuits: A Generalization of Monotone Boolean Circuits*

Rimli Sengupta[1] and H. Venkateswaran[2]

[1] Dept. of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803-3999
e-mail: rimli@cs.rose-hulman.edu
[2] College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
e-mail : venkat@cc.gatech.edu

Abstract. Cancellations are known to be helpful in efficient algebraic computation of polynomials over fields. We define a notion of cancellation in Boolean circuits and define Boolean circuits that do not use cancellation to be *non-cancellative*. Non-cancellative Boolean circuits are a natural generalization of monotone Boolean circuits. We show that in the absence of cancellation, Boolean circuits require super-polynomial size to compute the determinant interpreted over $GF(2)$. This non-monotone Boolean function is known to be in $\mathcal{P}$. In the spirit of monotone complexity classes, we define complexity classes based on non-cancellative Boolean circuits. We show that when the Boolean circuit model is restricted by withholding cancellation, $\mathcal{P}$ and popular classes within $\mathcal{P}$ are restricted as well, but classes $\mathcal{NP}$ and above remain unchanged.

1. Introduction

Using the power of cancellation to compute more efficiently has been a recurrent theme in the study of computational complexity. Strassen [15] made elegant usage of cancellation to obtain a surprising $O(n^{2.81})$ algorithm for matrix multiplication, improving the obvious $O(n^3)$ algorithm. Valiant [19] showed that usage of cancellations can lead to an exponential gain in size for counting the number of perfect matchings in triangular grid graphs. Nisan [10] showed that if cancellation is inhibited by withholding commutativity, then computing the determinant is as hard as computing the permanent. This paper investigates the power of cancellation in Boolean computation. We define a Boolean circuit to be *non-cancellative* if the formal polynomial associated with the circuit does not have a monomial that has both a literal and its complement. Non-cancellative circuits are a natural generalization of monotone Boolean circuits which have been studied extensively in the past [1, 4, 5, 11, 12, 16]. An important difference between the two models is that, unlike monotone Boolean circuits, non-cancellative circuits can compute all Boolean functions. This is because the circuit based on the representation of a Boolean function as the disjunction of its prime implicants is non-cancellative. All known lower bounds in the monotone circuit model carry over to the non-cancellative circuit model in a straightforward way. We also show that the lower bounds in the monotone circuit model can be adapted to similar lower bounds for a large class of non-monotone functions in the non-cancellative setting.

In the past, researchers have studied Boolean circuits with limited number of negations [9, 13, 17] to better understand the power of negations in Boolean computation. The study of non-cancellative circuits is of interest for the same reason. But in non-cancellative circuits the restriction is not on the number of negations but rather on the manner in which negations are used, namely, that they are not used for cancellation. This leads to the question as to whether non-trivial usage of negation is at all possible if cancellation is not allowed. The answer is yes. We have already observed that non-cancellative circuits can compute all Boolean functions. Moreover, there are examples of small non-cancellative circuits such as an $\mathcal{NC}^1$ circuit for PARITY.

This paper has three parts. In the first part, we generalize the lower bounds in the monotone Boolean circuit model to an appropriate class of non-monotone functions. We begin by showing that for every non-monotone function f such that $f(0) = 0$ or $f(1) = 0$, there is a monotone function g such that the non-cancellative complexity of f[3] is the same as the monotone complexity of g. We derive two important consequences of this result. First, in the context of computing monotone functions non-cancellative circuits are no more powerful than monotone circuits. This formalizes the intuition that for computing monotone functions, negations can only be used for cancellation. Thus, all lower bounds known for the monotone model apply in the non-cancellative model as well. The second consequence is that, for any monotone f, the non-cancellative complexity of the non-monotone function $\oplus f$ (see Sect. 3.2 for a definition) is at least the monotone complexity of f. This provides, for instance, a super-polynomial size lower bound for non-cancellative circuits that compute the determinant interpreted over $GF(2)$. This function is known to be in $\mathcal{P}$ [18]. This is the first example of a non-monotone Boolean function for which cancellations help.

In the second part of the paper, we quantify the amount of cancellation in a general non-monotone circuit. This quantity is defined to be the number of input variables that appear cancellatively in the formal polynomial associated with the circuit. We then show that non-monotone circuits in which $O(\log(n))$ variables appear cancellatively can be converted into an equivalent non-cancellative circuit without blowing up the size by more than a polynomial. This implies a lower bound on the number of variables that appear cancellatively in the formal polynomial of general Boolean circuits that compute certain monotone functions. For

* This work was supported by NSF grant CCR-9200878 and was done while the first author was at the College of Computing, Georgia Tech.

[3] The non-cancellative complexity of f is the size of the smallest non-cancellative circuit computing f.

instance, it follows that in the formal polynomial of any sublinear depth circuit for the bipartite perfect matching function at least a $o(n)$ of the input variables must appear cancellatively. Conversely, we have a linear depth lower bound on any circuit that computes the bipartite perfect matching function using $o(n)$ of the input variables cancellatively. This provides new insight into the role of cancellation in efficient computation of the perfect matching function; namely, in the context of depth requirement, allowing cancellations $o(n)$ input variables is as bad as withholding negations all together.

The third part of the paper is motivated by monotone complexity classes [5]. We define non-cancellative analogues of popular classes, such as $\mathcal{NL}$, $\mathcal{SAC}^1$ and $\mathcal{NP}$, using non-cancellative Boolean circuits. In addition to being a natural extension of the study of monotone complexity classes, the study of non-cancellative classes provides insight about the role of cancellation in structural issues, such as closure properties of complexity classes. For instance, while $\mathcal{NL}$ and $\mathcal{SAC}^1$ are known to be closed under complement [8, 3], we show that their non-cancellative analogues are not. As mentioned earlier, non-cancellative circuits can compute all Boolean functions, given enough size. It is therefore meaningful to ask how much size is necessary before cancellation becomes useless. To answer this, we show that the non-cancellative analogue of $\mathcal{NP}$ spans the whole of $\mathcal{NP}$. This implies that for all classes containing $\mathcal{NP}$, non-cancellativeness is not a restriction. However, we show that non-cancellativeness is a strict restriction for $\mathcal{P}$ and all popular classes within it.

Section 2 contains definitions and preliminary results used in the rest of the paper. The main parts of the paper are covered in Sections 3, 4 and 5. Concluding remarks are presented in Section 6.

2 Preliminaries

2.1 Definitions

Definition 2.1 A *Boolean circuit* B_n is a directed acyclic labeled graph in which each node has in-degree 0 or 2. Nodes with in-degree 0 are called *inputs* and are labeled from the set $\{x_i, \bar{x}_i, 0, 1 \mid 1 \leq i \leq n\}$. The other nodes (also called *gates*) are labeled from the set $\{\vee, \wedge\}$. The in-degree of a gate will be referred to as its *fan-in*. Every circuit has exactly one node with 0 out-degree and it is called the *output*. A *tree* is a Boolean circuit in which the gates have out-degree ≤ 1. It is called *monotone* if the inputs are labeled only from $\{x_i, 0, 1 \mid 1 \leq i \leq n\}$. It is called *non-monotone* if the inputs are labeled only from $\{\bar{x}_i, 0, 1 \mid 1 \leq i \leq n\}$. The *size* of a Boolean circuit is the number of gates in it and its *depth* is the length of the longest path from any input to the output.

Definition 2.2 Each element in the set $\{x_i \mid 1 \leq i \leq n\}$ is a *variable*. A *literal* is a variable in its positive form x_i or negative form $\bar{x}_i$. A *term* is a conjunction of literals, both positive and negative, and constants from $\{0, 1\}$. Each term t can be expressed as $c \cdot t_+ \cdot t_-$, where each literal in t_+ is positive and each literal in t_- is negative and $c \in \{0, 1\}$. For any term t, t_+ is the *positive term* of t and t_- is its *negative term*; $var(t_+)$ denotes the set of variables in t_+ and $var(t_-)$ the analogous set for t_-.

Each Boolean circuit B_n computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$. We shall use $\mathrm{PI}(f)$ to denote the set of prime implicants of a function f (see for instance [14] for a definition of prime implicants).

Definition 2.3 A *parse-graph* G of a Boolean circuit B_n is defined inductively as follows: G includes the output of B_n; for any $\vee$ gate v included in G, exactly one immediate predecessor of v in B_n is included as its only predecessor in G; and for any $\wedge$ gate v included in G, both the immediate predecessors of v in B_n are included as its predecessors in G.

Every gate v of a parse-graph G computes a term which is the conjunction of the labels on the inputs of the sub-graph rooted at v.

Definition 2.4 Let $\mathcal{G}$ denote the set of parse-graphs of B_n and let $t(G)$ be the term computed at the output of parse-graph G. The *formal polynomial* $P(B_n)$ of a circuit B_n is

$$\sum_{G \in \mathcal{G}} t(G)$$

Example 2.1 Consider the circuit B_3 in Fig. 1. The circuit is monotone and has eight parse-graphs. $P(B_3) = abc + a^2b + abc + ac^2 + a^2c + b^2c + ab^2 + abc$. Note that the prime implicants of the function computed by B_3 are ab, bc and ac. Thus, the number of monomials in the formal polynomial of a circuit is in general not the same as the number of prime implicants of the function that it computes.

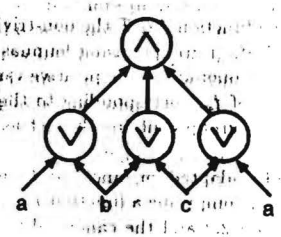

Fig. 1. The circuit B_3.

Definition 2.5 [4] A Boolean function f is said to be *positive monotone* (*negative monotone*) if the terms in $\mathrm{PI}(f)$ do not contain negative (positive, resp.) literals.

Definition 2.6 Let $\mathcal{F}$ be the class of Boolean functions such that $f(0) = 0$. For any $f \in \mathcal{F}$, we define the positive monotone function $f_m = \bigvee_{t \in PI(f)} t_+$. Similarly, let $\mathcal{F}$ be the class of Boolean functions such that $f(1) = 0$. For any $f \in \mathcal{F}$, we define the negative monotone function f_m as: $f_m = \bigvee_{t \in PI(f)} t_-$.

We shall henceforth say a function or a circuit is "monotone" to mean that it is either positive or negative monotone, in the style of [4]. Thus for a monotone function f, its complement $\bar{f}$ is monotone as well.

Definition 2.7 A term t is said to be *cancelled* if $var(t_+) \cap var(t_-) \neq \emptyset$. A term t is said to be *trivial* if it is cancelled or has 0 as its constant.

Definition 2.8 A Boolean circuit B_n is *non-cancellative* if there are no cancelled terms in its formal polynomial $P(B_n)$.

2.2 The Canonical Formal Polynomial

Given a general Boolean circuit B_n computing a function f such that $f(0) = 0$, we derive a canonical form for $P(B_n)$. A similar canonical form then follows for the dual class of functions for which $f(1) = 0$.

We first establish some relationships between the non-trivial monomials of $P(B_n)$ and the terms of $\mathrm{PI}(f_m)$ leading to a canonical form for $P(B_n)$. The proofs of the following lemmas are based on the idea that since B_n computes f, $P(B_n)$ and f must agree on every input assignment.

Lemma 2.1 For each non-trivial monomial ρ of $P(B_n)$, there exists a term $t \in \mathrm{PI}(f_m)$ such that $var(t) \subseteq var(\rho_+)$.

Lemma 2.2 For all terms $t \in \mathrm{PI}(f_m)$, there exists a non-trivial monomial ρ of $P(B_n)$, such that $var(t) = var(\rho_+)$.

For any B_n computing a function $f \in \mathcal{F}$ the non-trivial monomials of $P(B_n)$ can be partitioned into $|\mathrm{PI}(f_m)|$ groups, using lemmas 2.1 and 2.2, such that each group has at least one monomial whose positive variable set coincides with one of the prime implicants of f_m corresponding to the group and each of the rest of the monomials in the group contains this set as a subset of its positive variable set. formal

The above results can be adapted for functions f such that $f(1) = 0$. Given a general Boolean circuit B_n computing a function $f \in \mathcal{F}$, we have the following analogues of lemmas 2.1 and 2.2 and the canonical form for $P(B_n)$ follows in the manner similar to above.

Lemma 2.3 For each non-trivial monomial ρ of $P(B_n)$, there exists a term $t \in \mathrm{PI}(f_m)$ such that $var(t) \subseteq var(\rho_-)$.

Lemma 2.4 For all terms $t \in \mathrm{PI}(f_m)$, there exists a non-trivial monomial ρ of $P(B_n)$, such that $var(t) = var(\rho_-)$.

2.3 Functions Used

Here we define the functions used in this paper:

1. The bipartite perfect matching function BPM : $\{0,1\}^{n^2} \rightarrow \{0,1\}$, takes as input the standard $n \times n$ adjacency matrix representation of a bipartite graph G and outputs 1 if and only if G has a perfect matching. BPM is a monotone function.
2. The following function was considered by Tardos in [16], as an example for which the gap between the non-monotone and monotone complexity is exponential. TF: $\{0,1\}^{n^2} \rightarrow \{0,1\}$, takes as input the standard $n \times n$ adjacency matrix representation of a graph G and outputs 1 if and only if $\Phi(G) \leq f(n)$, where Φ, defined in [16], is a monotone graph property whose value lies between the clique number and chromatic number of G and f is any function such that $3 \leq f(n) \leq ((n/log(n))^{2/3})/4$. For the purpose of this paper, we choose $f(n) = ((n/log(n))^{2/3})/4$. TF is a monotone function.
3. $\oplus$BPM : $\{0,1\}^{n^2} \rightarrow \{0,1\}$ takes as input the standard $n \times n$ adjacency matrix representation of a bipartite graph G and outputs a 1 if and only if G has an odd number of perfect matchings. $\oplus$BPM is a non-monotone function. It is exactly the determinant function interpreted over $GF(2)$.
4. $\oplus$TF is defined analogously, by interpreting TF over $GF(2)$.

3 Non-cancellative vs. Monotone Complexity

Non-cancellative circuits are a strict generalization of monotone circuits since monotone circuits cannot compute non-monotone functions. In [11], Razborov had shown that monotone Boolean circuits are strictly weaker than general Boolean circuits for computing a monotone function: bipartite perfect matching. One interesting question is whether non-cancellative circuits are as powerful as general Boolean circuits. In Sect. 3.1 we show that non-cancellative circuits for monotone functions are no more powerful than monotone circuits. In Sect. 3.2 we exhibit a natural non-monotone function that is computable by polynomial size Boolean circuits but requires super-polynomial size non-cancellative circuits. To prove these results, we first show that for a large class of functions f, there exist monotone functions whose monotone complexity is similar to the non-cancellative complexity of f.

Based on the lemmas 2.1 and 2.2 and the canonical form presented in section 2.2, we can relate the non-cancellative complexity of any $f \in \mathcal{F}$ to the monotone complexity of f_m.

Theorem 3.1 For any function f such that $f(0) = 0$, if there is a non-cancellative circuit of size s and depth d computing f then there is a monotone circuit of size s and depth d computing f_m.

By the above theorem, any lower bound in the monotone model for a monotone function g applies to non-cancellative circuits that compute any function f such that $f_m = g$.

The non-cancellative complexity of any $f \in \bar{\mathcal{F}}$ can be analogously related to the monotone complexity of f_m, using lemmas 2.3 and 2.4.

Theorem 3.2 For any function f such that $f(1) = 0$, if there is a non-cancellative circuit of size s and depth d computing f then there is a monotone circuit of size s and depth d computing f_m.

3.1 Monotone Functions

Since monotone circuits are trivially non-cancellative, by applying Theorem 3.1 and 3.2 to monotone functions[4], we conclude that the monotone complexity and non-cancellative complexity of monotone functions are identical.

Theorem 3.3 For any monotone function f, there is a non-cancellative circuit of size s and depth d computing f if and only if there is a monotone circuit of size s and depth d computing f.

The above theorem formalizes the intuition that for computing monotone functions, negations can only be used for cancellation. As an immediate consequence of this theorem, known bounds in the monotone model [1, 16, 11, 12] apply for non-cancellative circuits.

Corollary 3.1 Non-cancellative circuits for BPM require size $n^{\Omega(\log n)}$ and depth $\Omega(n)$. Non-cancellative circuits for TF require size $2^{\Omega(n^{\frac{1}{3}-o(1)})}$.

The above corollary implies that the polynomial size circuits for BPM [7] and TF [16] must critically use the power of cancellation to compute these functions efficiently. Cancellations can therefore lead to exponential savings in size for computing monotone functions.

3.2 Non-monotone Functions

[illegible] are essential to computing non-monotone functions, but cancellations are not. In this section we show that non-cancellative circuits for the non-monotone function ⊕BPM can be no smaller than monotone circuits for BPM. The function ⊕BPM is exactly the determinant function interpreted over $GF(2)$ and is known to be in $\mathcal{P}$ [18].

We begin with the definition of the parity version of any Boolean function.

Definition 3.4 For any f, the function $\oplus f$ is defined as follows: $\oplus f$ outputs 1 on input x if and only if an odd number of prime implicants of f evaluate to 1 on input x.

[4] Except for the constant function $f = 1$, all monotone functions have $f(0) = 0$ or $f(1) = 0$. We say monotone functions to mean non-constant monotone functions.

For any function f, $\oplus f$ is non-monotone. The following facts are worth noting: (a) if $f(0) = 0$, then $\oplus f(0) = 0$; (b) for any monotone f, $(\oplus f)_m = f$, where $(\oplus f)_m$ is the monotone counterpart of $\oplus f$ as defined in Sect. 2.1. Therefore, for the class of functions $\{\oplus f \mid f \text{ is monotone}\}$ we have the following analogue of Theorem 3.3:

Theorem 3.4 For any monotone function f, if there is a non-cancellative circuit of size s and depth d computing $\oplus f$ then there is a monotone circuit of size s and depth d computing f.

As an immediate consequence of the above theorem and the known bounds for BPM in the monotone setting [16, 11, 12] we have,

Corollary 3.2 Non-cancellative circuits for ⊕BPM require size $n^{\Omega(\log n)}$ and depth $\Omega(n)$. Non-cancellative circuits for ⊕TF require size $2^{\Omega(n^{\frac{1}{3}-o(1)})}$ and depth $\Omega(n)$.

Since ⊕BPM is known to be in $\mathcal{P}$ [18], this demonstrates that cancellations can lead to super-polynomial savings in size even for computing non-monotone functions.

In summary, any polynomial size circuit for BPM or ⊕BPM must necessarily use cancellation. The result for BPM follows in a straightforward fashion from Razborov's lower bound [11]. The result for ⊕BPM is new.

4 Quantifying Cancellations

In this section, we quantify the amount of cancellations in a Boolean circuit and show that a circuit with a small amount of cancellations can be efficiently converted into a non-cancellative circuit. Throughout this section and in the next section we will use n to mean the number of input variables.

Definition 4.1 A variable x is said to *occur cancellatively* in B_n if there is a monomial $\rho \in P(B_n)$ such that $x \in var(\rho_+) \cap var(\rho_-)$. A circuit B_n is said to be *k-cancellative* if there are k variables that occur cancellatively in B_n, $0 \leq k \leq n$.

Theorem 4.1 Let B_n^k be a k-cancellative circuit of size s and depth d computing f, then there is an equivalent non-cancellative circuit of size $O(s2^k)$ and depth $(d+k)$, $0 \leq k \leq n$.

Proof: Let $P(B_n)$ have k variables $\{x_1, x_2, \cdots, x_k\}$ occurring cancellatively in B_n. We give a procedure to make the circuit non-cancellative with respect to one variable at a time, such that the circuit size is at most doubled at any step and the depth increases by 1. This, the resulting circuit, non-cancellative with respect to all the k variables, has size $O(s2^k)$ and depth $(d+k)$. We give the construction for x_1, the rest is immediate.

Let B_n^1 and B_n^2 be two copies of B_n such that in B_n^1 the leaf labeled x_1 is relabeled with z and in B_n^2 the leaf labeled x_1 is relabeled with $\bar{z}$. Consider the

circuit B'_n obtained by $\vee$-ing B^1_n and B^2_n and tying the leaf z to 0. It is easy to verify that B'_n is equivalent to B_n and that x_i does not occur cancellatively in B'_n. Also, the size of B'_n is $2s+1$ and its depth is $d+1$. □

Corollary 4.1 Non-cancellative circuits can compute f within polynomial size if and only if f is computable by $O(\log n)$-cancellative circuits within polynomial size.

4.1 Consequences

The construction in the above proof has interesting consequences in relating the size or depth of a circuit to the number of variables that appear cancellatively within it.

Theorem 4.2 If non-cancellative circuits are known to require size $\Omega(s)$ for a function f, then any circuit computing f within size $s' \leq s$ must be k-cancellative, for $k = \Omega(\log(\frac{s}{s'}))$.

Based on the non-cancellative size lower bounds for BPM and ⊕BPM in the previous section we have,

Corollary 4.3 Any polynomial size circuit for BPM or ⊕BPM must be k-cancellative for $k = \Omega(\log^2(n))$.

Since $s \cdot 2^k = \Omega(s)$ (in proof of Theorem 4.2), we also have the converse that any k-cancellative circuit for BPM or ⊕BPM with $k = o(\log^2 n)$, must have size $2^{\Omega(\log^2 n)}$. This result generalizes the size lower bounds in corollaries 3.1 and 3.2 and says that even allowing cancellations for $o(\log^2 n)$ variables is not enough to compute (even) a perfect matching within polynomial size.

All of the above results have depth analogues.

Theorem 4.3 If non-cancellative circuits are known to require depth $\Omega(d)$ for a function f, then any circuit computing f within depth $d' = o(d)$ must be k-cancellative, for $k = \Omega(d)$.

Using non-cancellative depth lower bounds for BPM and ⊕BPM in Section 3 we have the following corollary. Note that $\sqrt{n}$ appears in place of n since this section refers to the number of input variables.

Corollary 4.4 Any $o(\sqrt{n})$ depth circuit for BPM or ⊕BPM, must be k-cancellative for $k = o(\sqrt{n})$.

Or, in contrapositive, any non-cancellative circuit for BPM or ⊕BPM with $k = o(\sqrt{n})$ must have depth $\Omega(\sqrt{n})$. This result generalizes the depth lower bounds in corollaries 3.1 and 3.2 and implies that in the context of depth requirement of circuits computing the perfect matching, allowing cancellations on only $o(\sqrt{n})$ of the variables is as bad as withholding negations all together.

5 Non-cancellative Complexity Classes

Grigni [4] used monotone Boolean circuits to define monotone analogues of standard complexity classes such as $\mathcal{NC}^k$, $\mathcal{NL}$, $\mathcal{P}$ and $\mathcal{NP}$. In this section we consider non-cancellative analogues of standard complexity classes. In addition to being a natural extension of the study of monotone complexity classes, the study of non-cancellative classes is interesting because it lends insight into the role of cancellation in the context of structural issues, such as closure properties of complexity classes. As has been mentioned earlier, non-cancellative Boolean circuits can compute all functions given enough resources. A natural question is: what is the smallest class for which non-cancellativeness is not a restriction? Our result is that for $\mathcal{P}$ and popular classes within it, non-cancellativeness is a strict restriction, but not so for classes $\mathcal{NP}$ and above.

To define non-cancellative classes uniformly, we use the uniformity notions defined in [4] for monotone classes. A Boolean circuit is said to be *skew* if any $\wedge$-node has at most one gate as a predecessor. A Boolean circuit is said to have *semi-unbounded* fanin if any $\wedge$-node has fanin 2 but $\vee$-nodes could have unbounded fanin.

Definition 5.1 The following are classes of functions computable by uniform families of non-cancellative circuits within the indicated resources: (i) $\Diamond\mathcal{NL}$: skew polynomial size; (ii) $\Diamond\mathcal{NC}^k$: bounded fanin polynomial size and $O(\log^k(n))$ depth; (iii) $\Diamond\mathcal{SAC}^k$: semi-unbounded fanin polynomial size and $O(\log^k(n))$ depth; (iv) $\Diamond\mathcal{AC}^k$: unbounded fanin polynomial size and $O(\log^k(n))$ depth; (v) $\Diamond\mathcal{P}$: polynomial size; (vii) $\Diamond\mathcal{NP}$: skew polynomial depth.

As a direct consequence of theorem 3.3, we get the following separations between non-cancellative and non-monotone classes based on the analogous separations between monotone and non-monotone classes [5, 2, 11, 12] using monotone functions.

Theorem 5.1 $\Diamond\mathcal{AC}^0 \neq \mathcal{AC}^0$, $\Diamond\mathcal{NC}^k \neq \mathcal{NC}^k$, $\Diamond\mathcal{NL} \neq \mathcal{NL}$, $\Diamond\mathcal{SAC}^k \neq \mathcal{SAC}^k$, $\Diamond\mathcal{AC}^k \neq \mathcal{AC}^k$, $\Diamond\mathcal{P} \neq \mathcal{P}$.

The results in section 3 have other interesting consequences for complexity classes. For monotone classes that are known not to be closed under complement, we can prove the analogous non-closure results in the non-cancellative setting, using arguments similar to those above. Grigni [5] had shown that $m\mathcal{SAC}^1$ and $m\mathcal{NL}$ are not closed under complement.

Theorem 5.2 $\Diamond\mathcal{SAC}^1$ and $\Diamond\mathcal{NL}$ are not closed under complement.

The classes $\mathcal{NL}$ and $\mathcal{SAC}^1$ are both known to be closed under complement [8, 3]. The above result therefore implies that cancellations are critical to attain closure under complementation of these classes. For monotone classes that are known to be closed under complement, we do not know if the closure result also holds for the analogous non-cancellative class. For example, $m\mathcal{NC}^k$ and $m\mathcal{P}$ [1] are closed under complement by definition, but we cannot say the same about $\Diamond\mathcal{NC}^k$ and $\Diamond\mathcal{P}$.

3.1 $\mathcal{NP}$ is Non-cancellative

In [5], Grigni had shown that the monotone analogue of $\mathcal{NP}$ is exactly the class of monotone functions in $\mathcal{NP}$. We now show that the non-cancellative analogue of $\mathcal{NP}$ covers the whole of $\mathcal{NP}$. To prove this, we carry through the construction in the proof of Theorem 4.1 for all the variables. The resulting circuit has size $O(s2^n)$ and depth $d+n$, where n is the number of input variables. The construction in the proof of Theorem 4.1 is uniform and has the nice property that if the original circuit is skew, so is the final circuit. Therefore, based on the circuit definition of $\mathcal{NP}$ as the class of functions computable by uniform families of polynomial depth skew circuits [20], we have

Theorem 3.3 $\Diamond\mathcal{NP} = \mathcal{NP}$.

It follows that $\Diamond\mathcal{C} = \mathcal{C}$, for all known classes $\mathcal{C}$ containing $\mathcal{NP}$ that has a circuit definition in terms of size and/or depth.

Since non-cancellative circuits can compute all Boolean functions given enough resources, we had earlier posed the question of how much size is necessary before cancellation becomes useless. The above result shows that $\mathcal{NP}$ circuits are large enough for cancellation to become useless.

4 Summary and Open Questions

Non-cancellative Boolean circuits appear to be an appropriate generalization of monotone Boolean circuits such that all Boolean functions are computable. By defining a notion of cancellation in Boolean circuits, we study the power of negation at a finer granularity than previously considered and clarify the role of negation as that of cancellation. Previous work with monotone computations shows that cancellations are essential for efficiently computing certain monotone functions. The work presented in this paper extends that to the computation of non-monotone functions and shows that it is the cancellative aspect of negation that allows Boolean circuits to efficiently compute even certain non-monotone functions. We thereby give the first example of a non-monotone Boolean function for which cancellations help. We also show that cancellations are crucial for certain classes to be closed under complement, for example $\mathcal{NL}$. By showing $\mathcal{NP}$ is non-cancellative we identify the boundary at which cancellations are rendered powerless since in this sense it is the most unstructured of all languages in $\mathcal{NP}$.

This work raises several questions, here are a few: (i) In general, what can be said about the classes under complement of non-cancellative analogues of classes whose monotone versions are known to be closed under complement? (ii) Are there examples of $\mathcal{P}$-complete functions in $\Diamond\mathcal{P}$? (iii) Like $\mathcal{NP}$, does the non-cancellative analogue of $\mathcal{CO}-\mathcal{NP}$ span the whole of $\mathcal{CO}-\mathcal{NP}$? (iv) What kind of functions f such that the non-cancellative complexity of f is exponentially related to its non-monotone complexity? (v) Theorem 3.4 states that for any monotone f, the non-cancellative complexity of $\oplus f$ is bounded by the monotone complexity of f. Does the converse hold?

References

1. N. Alon and R.B. Boppana, *The monotone circuit complexity of Boolean functions*, Combinatorica, 7 (1987), 1-22.
2. M. Ajtai and Y. Gurevich, *Monotone versus positive*, J. Assoc. Comput. Mach. 34:4 (1987), 1004-1015.
3. A. Borodin, S. Cook, P. Dymond, W. Ruzzo, M. Tompa, *Two applications of inductive counting for complementation problems*, SIAM J. Comput., 18 (1989), 559-578.
4. M. Grigni, *Structure in monotone complexity*, Ph.D. thesis, M.I.T., 1991.
5. M. Grigni and M. Sipser, *Monotone separation of Logspace from NC*, Proc. 6th IEEE Structures (1991), 294-298.
6. M. Grötschel, L. Lovász and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1 (1981), 169-197.
7. J.E. Hopcroft and R.M. Karp, *A $n^{5/2}$ algorithm for maximum matching in bipartite graphs*, SIAM J. Comput. 2 (1973), 225-231.
8. N. Immerman, *Nondeterministic space is closed under complement*, SIAM J. Comput., 17 (1988), 935-938.
9. A. Markov, *On the inversion complexity of systems of Boolean functions*, Doklady 4:3 (1963), 694-696.
10. N. Nisan, *Lower bounds for non-commutative computation*, Proc. 23rd annual ACM STOC (1991), 410-418.
11. A.A. Razborov, *A lower bound on the monotone network complexity of the logical permanent*, Mathematischi Zametki 37 (1985), 887-900.
12. R. Raz and A. Wigderson, *Monotone circuits for matching require linear depth*, Proc. 22nd annual ACM STOC (1990), 287-292.
13. M. Santha and C. Wilson, *Polynomial size constant depth circuits with a limited number of negations*, Proc. 8th STACS (1991), 228-237.
14. J. E. Savage, *The complexity of computing*, R. E. Kreiger Publishing Co., Malabar, Florida, 1987.
15. V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. 13 (1969) 354-356.
16. É. Tardos, *The gap between monotone and non-monotone circuit complexity is exponential*, Combinatorica 7 (1987), 141-142.
17. K. Tanaka and T. Nishino, *On the complexity of negation-limited Boolean networks*, Proc. 26th ACM Symposium on Theory of Computing (1994), 38-47.
18. L. Valiant, *The complexity of computing the permanent*, Theoretical Computer Science 8 (1979), 189-201.
19. L. Valiant, *Negation can be exponentially powerful*, Theoretical Computer Science 12 (1980), 303-314.
20. H. Venkateswaran, *Circuit definitions of nondeterministic complexity classes*, SIAM J. Comput. 21 (1992) 655-670.

A Lower Bound for Monotone Arithmetic Circuits Computing 0-1 Permanent *

Rimli Sengupta†
Dept. of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803-3999
e-mail: rimli@cs.rose-hulman.edu

H. Venkateswaran
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
e-mail : venkat@cc.gatech.edu

June 19, 1996

Keywords: Permanent, size lower bound, arithmetic circuit.

1 Introduction

The permanent of an $n \times n$ matrix $X = [x_{ij}]$ is defined as follows:

$$\text{PERM}[X] = \sum_{\pi \in S_n} \prod_{i=1,n} x_{i,\pi(i)},$$

where S_n is the set of all permutation functions on n elements. In this note, we prove an exponential size lower bound for computing the permanent of matrices with only 0-1 entries (referred to as the 0-1 permanent here) using monotone arithmetic circuits. The permanent is believed to be much hard to compute and, in fact, proved by Valiant [5] to be NP-hard. In [1], Jerrum and Snir showed that algebraic circuits over certain semi-rings require exponential size to compute the permanent polynomial. In particular, their lower bound applies to the semi-ring of reals, with the usual multiplication and addition operators. Since there are no additive inverses, circuits over such an algebra can only compute "monotone" polynomials, that is, polynomials with positive coefficients. To reflect this, we refer to circuits over the semi-ring of reals as monotone arithmetic circuits. The

*This work was supported by NSF grant CCR-9200878.

†This work was done while the author was at the College of Computing, Georgia Tech, Atlanta GA 30332-0280.

lower bound in [1] does not apply to monotone arithmetic circuits computing the 0-1 permanent. To see why this is the case let us consider a simple example. Suppose we were interested in computing the polynomial function $f(x, y, z) = x \cdot y + y \cdot z$. If x, y, z could be any real number, then the above polynomial uniquely represents f. But if x, y, z are restricted to have 0-1 values, since 0 and 1 are idempotent with respect to the multiplication operator there are infinitely many representations of f: $x^{k_1} \cdot y^{k_2} + y^{k_3} \cdot z^{k_4}$, where k_1, k_2, k_3, k_4 are positive integers possibly all different. By a similar argument, the class of circuits that compute the 0-1 permanent function contains the circuits that compute the permanent function. This is why a circuit size lower bound for the latter does not apply to the former. We extend the framework in [1] to show that monotone arithmetic circuits require exponential size even for computing the 0-1 permanent. Showing that the above lower bound holds even when the Boolean complements of the inputs are supplied has interesting consequences for counting complexity classes (see section 5).

2 Definitions

A monotone arithmetic circuit A_n over the reals ($\mathcal{R}$) is a directed acyclic labeled graph in which nodes either have in-degree 0 or 2. Nodes with in-degree 0 are called *inputs* and are labeled from the set $\{x_i | \ 1 \leq i \leq n\}$. The other nodes (also called *gates*) are labeled from the set $\{+, \times\}$. The circuit has exactly one node with 0 out-degree and it is called the *output*. The *size* of an arithmetic circuit is the number of gates in it and its *depth* is the length of the longest path from any input to the output.

Each finite A_n computes a polynomial function $f : \mathcal{R}^n \rightarrow \mathcal{R}$. A *term* is a product of elements from the set $\{x_i | \ 1 \leq i \leq n\}$. We shall use $var(t)$ to denote the set of variables in a term t.

Definition 2.1 A *parse-graph* G of a circuit A_n is defined inductively as follows: G includes the output of A_n; for any $+$-node v included in G, exactly one immediate predecessor of v in A_n is included as its only predecessor in G; and for any $\times$-node v included in G, both the immediate predecessors of v in B_n are included as its predecessors in G.

Each parse-graph of a circuit A_n is rooted at the output gate of A_n. Every gate v of a parse-graph G computes a term which is the product of the labels on the inputs of the sub-graph rooted at v. With each A_n we associate a formal polynomial $P(A_n)$, which is the sum of the terms computed at the root of each parse-graph of A_n.

Definition 2.2 Let $\mathcal{G}$ denote the set of parse-graphs of A_n and let $t(G)$ be the term computed at the output of parse-graph G. The *formal polynomial* $P(A_n)$ of a circuit A_n is

$$\sum_{G \in \mathcal{G}} t(G).$$

We note the monomials of $P(A_n)$ have coefficients equal to $+1$.

3 Monotone Arithmetic Circuits for 0-1 Permanent

Consider the 0-1 permanent function 0-1PERM: $\{0,1\}^{n^2} \rightarrow \mathcal{N}$ defined as follows:

$$\text{0-1PERM}[X] = \sum_{\pi \in S_n} \prod_{i=1,n} x_{i,\pi(i)},$$

where $X = [x_{ij}]$ is an $n \times n$ matrix and S_n is the set of all permutation functions on n elements. Let p_π denote the term $\prod_{i=1,n} x_{i,\pi(i)}$, for $\pi \in S_n$. We will refer to each p_π as a *permutation.*

Throughout this section we will assume that A_{n^2} is a monotone arithmetic circuit computing 0-1PERM. We first establish a canonical form for $P(A_{n^2})$. The proofs of the following lemmas are based on the idea that since A_{n^2} computes 0-1PERM, $P(A_{n^2})$ and 0-1PERM must agree on every input assignment.

Lemma 3.1 For each monomial ρ in $P(A_{n^2})$, $var(p_\pi) \subseteq var(\rho)$ for some $\pi \in S_n$.

Proof: Suppose there is a monomial ρ in $P(A_{n^2})$ such that there is no $\pi \in S_n$ with $var(p_\pi) \subseteq var(\rho)$. On the input assignment that sets the variables in $var(\rho)$ to 1 and the rest to 0, 0-1PERM evaluates to 0 but $P(A_{n^2})$ evaluates to 1, leading to a contradiction. □

Lemma 3.2 For all $\pi \in S_n$, there exists a $\rho \in P(A_{n^2})$ such that $var(\rho) = var(p_\pi)$.

Proof: Suppose there is a $\pi \in S_n$ such that there is no monomial ρ in $P(A_{n^2})$ with $var(\rho) = var(p_\pi)$. Consider the input assignment that sets the variables in $var(p_\pi)$ to 1 and all the rest to 0. 0-1PERM evaluates to 1 on this input. For $P(A_{n^2})$ to be non-zero on this input, there must exist a monomial ρ in $P(A_{n^2})$ such that $var(\rho) \subset var(p_\pi)$. By lemma 3.1, this implies the existence of a $\sigma \in S_n$ such that $var(p_\sigma) \subset var(p_\pi)$, which is impossible. □

Based on the above lemmas, $P(A_{n^2})$ has at least $n!$ monomials, one for each p_π, $\pi \in S_n$. If it had any more, then on the input that assigns all variables to 1, 0-1PERM evaluates to $n!$ but $P(A_{n^2})$

would evaluate to a value strictly greater than $n!$. Thus, the above lemmas completely determine the variable sets of each monomial in $P(A_{n^2})$ and we have,

Theorem 3.1 For any monotone arithmetic circuit A_{n^2} computing 0-1PERM,

$$P(A_{n^2}) = \sum_{\pi \in S_n} \prod_{i=1}^{n} x_i^{k_i},$$

where each k_i is a natural number.

In [1], Jerrum and Snir prove an exponential lower bound on the size of monotone arithmetic circuits A_{n^2} that compute the permanent function over the reals. That is,

$$P(A_{n^2}) = \sum_{\pi \in S_n} \prod_{i=1}^{n} x_i.$$

In the next section we extend their framework to show that a similar lower bound holds for monotone arithmetic circuits that compute the 0-1 permanent function. The main difference between this result and the result in [1] is that the formal polynomial associated with the circuits used here need not be multi-linear.

4 Adaptation of Jerrum and Snir's Framework

Let A_{n^2} be a monotone arithmetic circuit computing 0-1PERM. Throughout this section, we will use the fact that A_{n^2} has exactly $n!$ parse-graphs, each computing a permutation term p_π that has exactly n variables.

Definition 4.1 For a $\times$-node α, let $m(\alpha)$ be the number of parse-graphs of A_{n^2} in which α appears.

Definition 4.2 A $\times$-node α is said to be *(r,d)-significant* for $1 \leq r \leq n$ and $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$, if α appears in a parse-graph that computes a term with r variables, exactly d of which are contributed by one of the immediate predecessors of α alone.

If a $\times$-node α is not (r, d)-significant for any (r, d), then $m(\alpha) = 0$. Moreover, as a part of the proof of lemma 4.1 below, we show that α cannot be (r, d)-significant for more than one (r, d) pair.

Definition 4.3 Let H be a subgraph of a parse-graph G. Define the *weight* of H as follows: $W(H) = \sum_{\alpha \in \times-nodes(H)} \frac{1}{m(\alpha)}$, where $\times$-*nodes*(H) denotes the set of $\times$-nodes in H.

A lemma similar to the one below was proven in [1] for circuits all of whose parse-graphs are trees. We show that the lemma holds even for circuits whose parse-graphs are not necessarily trees.

Lemma 4.1 If α is an (r, d)-significant $\times$-node of A_{n^2}, then $m(\alpha) \leq d!(r-d)!(n-r)!$.

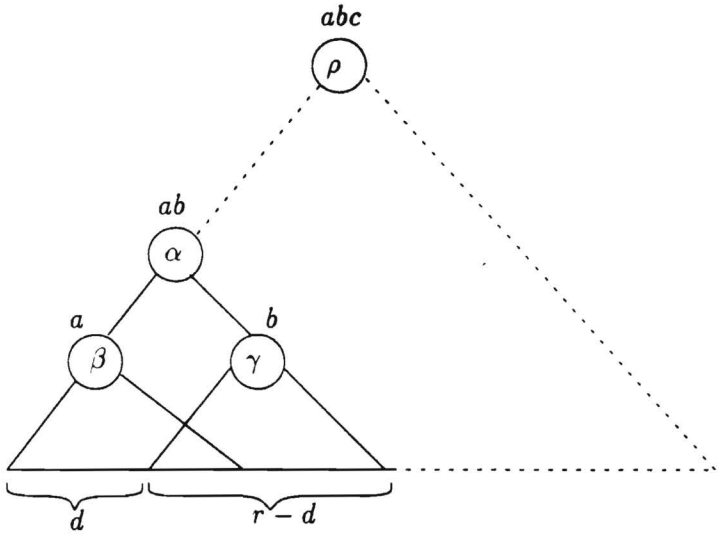

Figure 1: (r,d)-significant node α embedded in circuit A_{n^2} with output node ρ.

Proof: Let β and γ be the immediate predecessors of α in A_{n^2}. Let G be a parse-graph in which α appears with an r-variable term. Let a be the term formed at β and b be the term formed at γ such that $a \cdot b$ has $r \geq 1$ variables and a has $0 \leq d \leq \lfloor r/2 \rfloor$ variables that are not in b (see figure 1). Let c be a term such that $a \cdot b \cdot c$ is the n-variable term formed at the root of G. Clearly, c has $n - r$ variables that are not in a or b. Note that the sets $var(a)$, $var(b)$ and $var(c)$ may have non-empty intersections since G is a parse-graph as opposed to a parse-tree.

Let α participate in another parse-graph G'. Since α is a $\times$-node, β and γ participate in G' as well. In G', let a' be the term formed at β and b' be the term formed at γ. Let $a' \cdot b' \cdot c'$ be the term formed at the root of G', which by theorem 3.1 must be different from the term $a \cdot b \cdot c$ computed by G. Moreover, since α is a $\times$-node, there must be parse-graphs in A_{n^2} that compute the rest of the terms in the product $(a + a')(b + b')(c + c')$. But by theorem 3.1, every term computed by a parse-graph of A_{n^2} must be a permutation. Thus, the number of parse-graphs in which α participates is the number of distinct triples (a, b, c) where a is a term formed at β, b is a term formed at γ and c is a term such that $a \cdot b \cdot c$ is a permutation.

We first show that for a fixed r and d, $m(\alpha)$ cannot exceed $d!(r-d)!(n-r)!$. Let

$$\begin{aligned} A &= var(a) - var(b \cdot c) \\ B &= var(b) - var(a \cdot c) \\ C &= var(c) - var(a \cdot b) \end{aligned}$$

It is easy to see that α can participate in any parse-graph that computes the term $a' \cdot b' \cdot c'$, where the row (column) index of any variable in a' is the row (column, resp.) index of some variable in A and b', c' are obtainable similarly from B, C, respectively. In other words, the indices of the variables in a' (b', c') are can be obtained by permuting within the indices of the variables in A (B, C resp.). Therefore, α can participate in $|A|!\,|B|!\,|C|!$ parse-graphs. Clearly, $|A|!\,|B|!\,|C|! \leq d!\,(r-d)!\,(n-r)!$.

Suppose α participates in a parse-graph G' different from the $|A|!|B|!|C|!$ parse-graphs described above such that in G', β computes the term a', γ computes the term b' and the output node computes the term $a' \cdot b' \cdot c'$. By the choice of G', $a' \cdot b' \cdot c'$ must be a permutation different from the $|A|!\,|B|!\,|C|!$ permutations above. Thus, the indices of the variables in at least one of a', b' or c' are not obtainable by permuting within the indices of the variables in A, B or C, respectively. Without loss of generality, suppose this is true for a' (the argument is symmetrical for b' or c'). Then, there must be a variable x_{ij} in $var(a')$ such that either i is not the row index of any variable in A, or j is not the column index of any variable in A or both. But then, the term $a' \cdot b \cdot c$ is not a permutation and hence cannot be a monomial of $P(A_{n^2})$, by theorem 3.1. Therefore, G' cannot exist. Note that we did not make any assumptions about a' or b' other than that $a' \cdot b' \cdot c'$ is a permutation different from those counted earlier. Therefore, this argument holds even if $|var(a' \cdot b')| = r'$ and $|var(a') - var(b')| = d'$, where $r' \neq r$ and $d' \neq d$. Since this means that the node α contributes to $m(\alpha)$ only for a single value of r and a single value of d, $m(\alpha)$ is bounded above by $d!(r-d)!(n-r)!$. □

Let $\{G_i \mid 1 \leq i \leq n!\}$ be the parse-graphs of A_{n^2}. Let $\mathcal{X} = \{\times\text{-node } \alpha \mid m(\alpha) \geq 1\}$. The lemma below is motivated by theorem 3.3 in [1].

Lemma 4.2 $\sum_{i=1}^{n!} W(G_i) = |\,\mathcal{X}\,|$.

Proof: By definition,

$$\sum_{i=1}^{n!} W(G_i) = \sum_{i=1}^{n!} \sum_{\alpha \in \times\text{-nodes}(G_i)} \frac{1}{m(\alpha)}.$$

Fix an $\times$-node α. For each parse-graph G_i, the contribution by α to the sum on the right-hand side of the above equation is either 0 (if α does not occur in it) or $\frac{1}{m(\alpha)}$. Thus, the total contribution by α is $m(\alpha)\frac{1}{m(\alpha)} = 1$ and therefore the right-hand side is the number of $\times$-nodes in $\mathcal{X}$. □

To obtain a lower bound on the weight of a parse-graph G, we consider the number of input variables covered by G, instead of the notion of degree used in [1]. This is because $P(A_{n^2})$ is not

necessarily multi-linear in our model.

For any parse-graph G define a *sub-parse-graph* of G to be any parse-graph H (as in definition 2.3) rooted on any node of G. For any sub-parse-graph H of a parse-graph of A_{n^2}, let $v(H)$ denote the number of variables in the term computed at the root of H.

Let $c(r,d) = d!(r-d)!(n-r)!$. The lemma below is adapted from theorem 3.4 in [1].

Lemma 4.3 For any sub-parse-graph H of any parse-graph G of A_{n^2}, $W(H) \geq \sum_{i=2}^{v(H)} \frac{1}{c(i,1)}$.

Proof: The proof is by induction on the number of nodes in H. For the base case, H has a single node. Since it must be an input node, $v(H) = 1$, and since H has no $\times$-nodes, $W(H) = 0$. Thus, the lemma holds.

For the induction step, let α be the root node of H and let $v(H) = r$. Without loss of generality, we assume that α is a $\times$-node[1]. Since α appears in G, it must be (r,d)-significant for some d, $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$. Let α have immediate predecessors β and γ, with β contributing d variables alone (see figure 1). Let H_γ be the sub-parse-graph of G rooted at γ. Clearly, $v(H_\gamma) = (r-d)$. For each node x in the set of d input nodes in figure 1, there is a path P_x from x to β that is disjoint from H_γ. Let $\tilde{H}_\beta$ be the edge-induced subgraph of H_β defined on the union of the edge sets of P_x, for all x. The graph $\tilde{H}_\beta$ is thus a subgraph of G rooted at β such that $v(\tilde{H}_\beta) = d$. Since $\tilde{H}_\beta$ and H_γ have no $\times$-nodes in common we have,

$$W(H) \geq W(\tilde{H}_\beta) + W(H_\gamma) + \frac{1}{m(\alpha)}.$$

Note that $m(\alpha) \geq 1$ since α appears in G. Applying the inductive hypothesis to the subgraphs $\tilde{H}_\beta$ and H_γ and using lemma 4.1, we get:

$$W(H) \geq \sum_{i=2}^{d} \frac{1}{c(i,1)} + \sum_{i=2}^{(r-d)} \frac{1}{c(i,1)} + \frac{1}{c(r,d)}, \quad \text{for some } d, \quad 0 \leq d \leq \lfloor \frac{r}{2} \rfloor.$$

Let the expression on the right be denoted $\Phi(d)$. In the range $1 \leq d \leq \lfloor \frac{r}{2} \rfloor$ $\Phi(d)$ is shown to be minimum at $d = 1$ [1]. But $\Phi(0) > \Phi(1)$, since $\frac{1}{c(r,0)} \geq 0$. Therefore, $\Phi(d) \geq \Phi(1)$, for all d in the range $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$. The lemma then follows from the facts that $W(H) \geq \Phi(d)$, for some d, $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$ and $\Phi(1) = \sum_{i=2}^{r} \frac{1}{c(i,1)}$. □

The above lemmas lead to the lower bound for 0-1PERM using monotone arithmetic circuits.

Theorem 4.1 Any monotone arithmetic circuit A_{n^2} requires size $\geq n(2^{n-1} - 1)$ to compute 0-1PERM.

[1] Otherwise consider the first $\times$-node on any path starting at the root and going towards the input nodes.

Proof: From lemmas 4.2 and 4.3 it follows that the size of A_{n^2} for this problem is at least $\sum_{j=1}^{n!} \sum_{i=2}^{v(G_j)} \frac{1}{c(i,1)}$. But $v(G_j) = n$ since every parse-graph of A_{n^2} computes a permutation. Therefore, the above expression is equivalent to $n! \sum_{i=2}^{n} \frac{1}{c(i,1)}$. Substituting for $c(i,1)$ in $\sum_{i=2}^{n} \frac{1}{c(i,1)}$ we get, $\sum_{i=2}^{n} \frac{1}{(n-i)!\,(i-1)!}$ which is exactly $\frac{2^{n-1}-1}{(n-1)!}$, from which the theorem follows.□

5 Discussion

In this note, we proved an exponential lower bound on the size of monotone arithmetic circuits that compute the 0-1 permanent function. It would be interesting to generalize this result to arithmetic circuits, which are defined exactly as monotone arithmetic circuits except that the inputs are labeled from $\{x_i \mid 1 \leq i \leq n\} \cup \{-1\}$. That is, negations are available.

Another generalization of monotone arithmetic circuits is of greater interest in the context of complexity classes. Define a *counting* arithmetic circuit A_n similarly to a monotone arithmetic circuit except that the inputs are labeled from $\{x_i, (1 - x_i), 0, 1 \mid 1 \leq i \leq n\}$, where each x_i has a 0-1 value. That is, the Boolean complements of the input variables are also available. Such circuits only compute functions of the form $f : \{0,1\}^* \rightarrow \mathcal{N}$.

One reason that counting arithmetic circuits are of interest is because there exist characterizations of popular counting classes such as $\sharp\mathcal{P}$ and $\sharp\mathcal{SAC}^1$ in terms of these circuits [6, 7]. Two such characterizations are stated in the theorem below. Here, the size and depth of counting arithmetic circuits are defined as before and the uniformity notion is a standard one [3, 6]. The degree of a counting arithmetic circuit is the algebraic degree of its formal polynomial.

Theorem 5.1 [6] $\sharp\mathcal{P}$ is the class of functions computable by uniform families of counting arithmetic circuits within polynomial depth and polynomial degree. $\sharp\mathcal{SAC}^1$ is the class of functions computable by uniform families of counting arithmetic circuits within polynomial size and polynomial degree.

Therefore, extending the lower bound in this paper to hold for counting arithmetic circuits would have the interesting consequence that 0-1PERM $\notin \sharp\mathcal{SAC}^1$. This is turn would imply that $\sharp\mathcal{SAC}^1$ is properly contained in $\sharp\mathcal{P}$, since 0-1PERM is known to be in $\sharp\mathcal{P}$ [5].

Another reason that counting arithmetic circuits are of interest is due to the following observation. Let $\mathcal{PH}$ denote the polynomial-time hierarchy and $\mathcal{P}/poly$ denote the class of languages accepted by polynomial size Boolean circuits with polynomial length advice.

Theorem 5.2 If 0-1PERM is computable by polynomial size counting arithmetic circuits then $\mathcal{PH}$ collapses.

Proof: Let $\mathcal{A}$ be a polynomial size counting arithmetic circuit for 0-1PERM with n inputs. Since the inputs to $\mathcal{A}$ are from $\{0,1\}$, its output is guaranteed to be $\leq n!$. So, to keep the numbers computed at intermediate nodes of $\mathcal{A}$ from growing too large, pick a prime $p > n!$ and perform the $+$ and $\times$ operations in $\mathcal{A}$ modulo p. Now, by replacing the $+$ and $\times$ nodes in $\mathcal{A}$ with the appropriate Boolean circuits, and by providing the prime p as advice, it can be concluded that 0-1PERM can be computed by polynomial size Boolean circuits with polynomial length advice. Then, from Toda's result [4] that $\mathcal{PH} \subseteq \mathcal{P}^{\sharp\mathcal{P}}$, it follows that $\mathcal{PH} \subseteq \mathcal{P}/poly$. Such an inclusion implies the collapse of $\mathcal{PH}$, as shown by Karp and Lipton [2]. □

References

[1] M. Jerrum and M. Snir, Some exact complexity results for straight-line computations over semi-rings, *J. Assoc. Comput. Mach.* **29** (1982) 874-897.

[2] R.M. Karp and R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in: *Proc. 12th Annual Symp. on Theory of Computing* (1980) 302-309.

[3] W. L. Ruzzo, On uniform circuit complexity, *J. Comput. System Sci.* **22** (1981) 365-383.

[4] S. Toda, PP is as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **20** (1991) 865-877.

[5] L. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* **8** (1979) 189-201.

[6] H. Venkateswaran, Circuit definitions of nondeterministic complexity classes, *SIAM J. Comput.* **21** (1992) 655-670.

[7] V. Vinay, Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits, *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory* (1991) 270-284.

Multilinearity Can Be Exponentially Expensive *

Rimli Sengupta and H. Venkateswaran

College of Computing

Georgia Institute of Technology

Atlanta, GA 30332-0280

e-mail : {rimli,venkat}@cc.gatech.edu

March 8, 1996

*This work was supported by NSF grant CCR-9200878.

Abstract

We define a Boolean circuit to be *multilinear* if the formal polynomial associated with it is multilinear. We consider the problem of computing the connectivity function using circuits that are monotone and multilinear. Our main result is that monotone multilinear circuits for this function require exponential size. Since connectivity can be computed by monotone Boolean circuits within size $O(n^3)$, the lower bound establishes that the multilinearity restriction can be exponentially expensive. Moreover, based on the observation that connectivity can be computed by monotone circuits of polynomial size and polynomial degree, the lower bound exhibits an exponential gap between multilinearity and polynomial degree.

1 Introduction

Every finite Boolean function can be represented as a multilinear polynomial. However, since the elements of Boolean algebra are idempotent with respect to the multiplicative operator, not all formal polynomials representing a Boolean function need be multilinear. We define a Boolean circuit to be *multilinear* if the formal polynomial associated with it is multilinear. Multilinearity is thus a restriction on the Boolean model, but unlike monotonicity it is not a restriction on the set of computable functions. The study of restricted models of Boolean computation has been largely motivated by the difficulty in obtaining non-trivial size or depth lower bounds for computing functions using general Boolean circuits. Monotonicity is an example of such a restriction for which several interesting results have been obtained [5, 10, 9, 15]. Just as the monotonicity restriction allows a study of the power of negation in Boolean computation, the multilinearity restriction allows a study of the power of multiplicative idempotence.

There are natural functions that multilinear circuits can compute within polynomial size, such as parity and Boolean matrix multiplication. Also, the polynomial size circuit [17] based on the Cocke-Kasami-Younger (see [6]) algorithm for context-free language recognition is multilinear.

We consider the problem of computing the connectivity function using circuits that are monotone and multilinear. The connectivity function has been studied extensively in the past [18]. Our main result is that monotone multilinear circuits for this function require exponential size. There is a monotone circuit based on the well-known algorithm due to Floyd and Warshall (see [1]) for transitive closure that computes connectivity within size $O(n^3)$. Our lower bound for connectivity establishes that in the context of monotone computation the multilinearity restriction can be exponentially expensive. Moreover, we observe that connectivity can be computed by monotone circuits of polynomial size and polynomial degree. The lower bound then implies an exponential gap between multilinearity and polynomial degree, in monotone Boolean circuits.

To prove our lower bound we build on a combinatorial framework developed by Jerrum and Snir [7] to obtain exponential size lower bounds for computing certain multilinear polynomials, including the spanning tree polynomial, using algebraic circuits over positive reals. We first extend this framework to show that the an exponential size bound holds for Boolean circuits whose formal polynomial is a linear combination of the terms in the spanning tree polynomial. The lower bound then follows from an efficient construction (lemma 3.4) that converts a monotone multilinear circuit for connectivity into one for which the above bound applies.

The rest of the paper is organized as follows. We begin by showing examples of natural functions that have efficient multilinear circuits (section 2.1). In section 2.2, we derive a canonical form for the formal polynomial of a monotone Boolean circuit computing a monotone function. In section 3, we adapt the Jerrum and Snir [7] lower bound framework to monotone multilinear circuits and obtain an exponential size lower bound for the connectivity function.

2 Preliminaries

Definition 2.1 A *Boolean circuit* B_n is a rooted directed acyclic labeled graph in which nodes either have in-degree 0 or 2. Nodes with in-degree 0 are called *inputs* and are labeled from the set $\{x_i, \bar{x}_i, 0, 1 | \ 1 \leq i \leq n\}$. The other nodes, also called *gates* are labeled from the set $\{\vee, \wedge\}$. The circuit has exactly one node with out-degree 0 and it is called the *output*. A *formula* is a Boolean circuit in which all the gates have out-degree ≤ 1. B_n is said to be *monotone* if the inputs are labeled from the set $\{x_i, 0, 1 | \ 1 \leq i \leq n\}$. The *size* of a Boolean circuit is the number of gates in it and its *depth* is the length of the longest path from any input to the output.

Definition 2.2 A *term* is a conjunction of variables. Let $var(t)$ denote the set of variables in term t.

Each B_n computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$. We shall use $\mathrm{PI}(f)$ to denote the set of prime implicants of f (see for instance [13] for a definition of prime implicants).

Definition 2.3 A *parse-graph* G in B_n is defined inductively as follows: G includes the output of B_n; for any $\vee$ gate v included in G, exactly one immediate predecessor of v in B_n is included as its only predecessor in G; and for any $\wedge$ gate v included in G, both the immediate predecessors of v in B_n are included as its predecessors in G.

Every node v of a parse-graph G computes a term which is the conjunction of the labels on the inputs of the sub-graph rooted at v. Let $t(G)$ be the term computed at the root of G and let $\mathcal{G}$ denote the set of parse-graphs of B_n.

Definition 2.4 The *formal polynomial* $P(B_n)$ of a circuit B_n is

$$\sum_{G \in \mathcal{G}} t(G).$$

Thus, each monomial of $P(B_n)$ corresponds to a parse-graph in B_n.

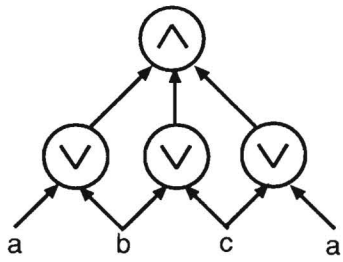

Figure 1: The circuit B_3.

Example 2.1 Consider the circuit B_3 in figure 1. The circuit is monotone and has eight parse-graphs. $P(B_3) = abc + a^2b + ac^2 + a^2c + b^2c + ab^2 + bc^2 + abc$. Note that the prime implicants of the function computed by B_3 are ab, bc and ac.

Definition 2.5 The *degree* of B_n is defined inductively: the degree of input labeled with a literal (constant) is 1 (resp. 0); the degree of a $\vee$ gate is the maximum of the degree of its two predecessors; the degree of a $\wedge$ gate is the sum of the degree of its two predecessors. The degree of B_n is the degree of its output.

Recall that a multivariate polynomial is multilinear if each variable occurs at most once in each monomial.

Definition 2.6 A Boolean circuit B_n is said to be *multilinear* if $P(B_n)$ is multilinear.

It can be verified that $P(B_n)$ is multilinear if and only if all the parse-graphs of B_n are trees. We shall use the term parse-trees instead of parse-graphs when considering multilinear circuits.

An immediate question is how do multilinear circuits compare with formulas? While formulas are necessarily trees, the inputs may have out-degree more than 1 and therefore the formal polynomial associated with a formula need not be multilinear. Conversely, a multilinear circuit is not necessarily a tree and hence not a formula. Given enough size, every Boolean function can be implemented either as a formula or a multilinear circuit. So to make the above question interesting, we need to impose resource bounds. The question then becomes whether polynomial size multilinear circuits compute $\mathcal{NC}^1$ functions. In the next section we show that there are natural $\mathcal{NC}^1$ functions that are computable by multilinear circuits within polynomial size. Moreover, there are also examples of functions computable by polynomial size multilinear circuits that are unlikely to be in $\mathcal{NC}^1$. We note that multilinear circuits can be no more powerful than polynomial degree circuits. It is known that polynomial degree circuits of polynomial size exactly define the class $\mathcal{SAC}^1$ [3, 16].

2.1 Efficent Multilinear Circuits for Natural Functions

The depth two circuit based on the representation of a Boolean function as a disjunction of its prime implicants is multilinear. Therefore, every Boolean function can be implemented multilinearly. However, such circuits are usually large since any non-trivial function has exponentially many prime implicants. In this section, we show that there are natural functions that can be computed using multilinear circuits within small size.

Multilinear Circuit for Parity We first show that there is an $\mathcal{NC}^1$ circuit for PARITY that is multilinear.

Lemma 2.1 PARITY_n and $\overline{\text{PARITY}}_n$ can be computed with $\mathcal{NC}^1$ circuits that are multilinear.

Proof: The proof is by induction on n.

For the base case, let $n = 2$. The circuits C_2 and $\bar{C}_2$ based on the expressions below compute $\text{PARITY}_2(x_1, x_2)$ and $\overline{\text{PARITY}}_2(x_1, x_2)$, respectively.

$$C_2 = (x_1 \wedge \bar{x_2}) \bigvee (\bar{x_1} \wedge x_2)$$

$$\bar{C}_2 = (x_1 \wedge x_2) \bigvee (\bar{x_1} \wedge \bar{x_2})$$

These circuits are easily seen to be multilinear. By inductive hypothesis, there exist multilinear circuits $C_{\frac{n}{2}}, \bar{C}_{\frac{n}{2}}, D_{\frac{n}{2}}$ and $\bar{D}_{\frac{n}{2}}$, such that $C_{\frac{n}{2}}$ $(\bar{C}_{\frac{n}{2}})$ computes $\text{PARITY}_{\frac{n}{2}}(x_1, \ldots, x_{\frac{n}{2}})$ $(\overline{\text{PARITY}}_{\frac{n}{2}}(x_1, \ldots, x_{\frac{n}{2}})$, resp.) and $D_{\frac{n}{2}}$ $(\bar{D}_{\frac{n}{2}})$ computes the same functions for the variable set $\{x_{\frac{n}{2}+1}, \ldots, x_n\}$. The circuits C_n $(\bar{C}_n)$ based on the expressions below compute PARITY_n and $\overline{\text{PARITY}}_n$, respectively.

$$C_n = (C_{\frac{n}{2}} \wedge \bar{D}_{\frac{n}{2}}) \bigvee (\bar{C}_{\frac{n}{2}} \wedge D_{\frac{n}{2}})$$

$$\bar{C}_n = (C_{\frac{n}{2}} \wedge D_{\frac{n}{2}}) \bigvee (\bar{C}_{\frac{n}{2}} \wedge \bar{D}_{\frac{n}{2}})$$

To show that C_n is multilinear, we observe that although the inputs sets of $C_{\frac{n}{2}}$ and $\bar{C}_{\frac{n}{2}}$ are the same, subgraphs of $C_{\frac{n}{2}}$ and $\bar{C}_{\frac{n}{2}}$ do not appear in the same parse-graph in C_n. Moreover, subgraphs of $C_{\frac{n}{2}}$ and $\bar{D}_{\frac{n}{2}}$ do appear in the same parse-graph in C_n but their input sets are disjoint. Multilinearity of C_n then follows by inductive hypothesis. The analysis for $\bar{C}_n$ is similar. It is easy to verify that C_n and $\bar{C}_n$ are both $\mathcal{NC}^1$ circuits.□

There are other $\mathcal{NC}^1$ functions that have polynomial size multilinear circuits. For example, adding 2 n-bit numbers as well as Boolean multiplication of 2 $n \times n$ matrices can be computed multilinearly within polynomial size.

Multilinear Circuit for CFL Membership We now show that the circuit implementation in [17] of the standard Cocke-Kasami-Younger algorithm for context-free language recognition is multilinear. This problem is known to be complete for $\mathcal{SAC}^1$ [11] and is therefore unlikely to be in $\mathcal{NC}^1$.

For a fixed context-free grammar in Chomsky normal form, a Boolean circuit family $\{G_n\}$ that accepts the language generated by the grammar can be derived from the Cocke-Kasami-Younger algorithm [6]. We show that this circuit implementation [17] is multilinear.

In what follows, x_{ij} denotes the substring $x_i x_{i+1} \ldots x_j$ of the input string $x_1 x_2 \ldots x_n$, $1 \leq i < j \leq n$. A gate in the circuit G_n has one of the following labels [17]:

- $[A, i, j]$, for some nonterminal A and integers i and j such that $1 \leq i \leq j \leq n$. This is an $\vee$ gate that evaluates to 1 on input x if and only if $A \stackrel{*}{\Rightarrow} x_{ij}$.

- $[B, C, i, j, k]$, for some integers i and j such that $1 \leq i \leq j \leq n$, for all pairs of nonterminals B and C for which $A \rightarrow BC$ is a production for some nonterminal A, and for some integer k such that $i \leq k < j$. This is an $\wedge$ gate that has two inputs $[B, i, k]$ and $[C, k+1, j]$ and it evaluates to 1 on input x if and only if $B \stackrel{*}{\Rightarrow} x_{ik}$ and $C \stackrel{*}{\Rightarrow} x_{k+1,j}$.

The output gate is $[S, 1, n]$ where S is the start symbol in the grammar G. The input to a gate of the form $[A, i, j]$ are for $i < j$ all gates of the form $[B, C, i, j, k]$ where $i \leq k < j$ and $A \rightarrow BC$ is a production in the grammar. The gate $[A, i, i]$ has a single input which is one of the following: (a) the constant 1 if both the productions $A \rightarrow 0$ and $A \rightarrow 1$ are in the grammar, and the constant 0 if neither production is in the grammar; (b) x_i (the ith input) if $A \rightarrow 1$ is a production in the grammar; (c) $\bar{x}_i$, if $A \rightarrow 0$ is a production in the grammar.

Lemma 2.2 The interval indicated in the label of any $\vee$ gate in the circuit rooted at β is a subinterval of $[i, k]$.

Proof: We assume that G_n is layered, without any loss of generality, with the inputs at level 0. The proof is by induction on level h of the circuit rooted at β, $0 \leq h \leq t$, β being at level t.

For the base case $h = t$, $[i, k]$ is a subinterval of itself.

By the inductive hypothesis, all $\vee$ gates at level h have labels marked with a subinterval of $[i, k]$.

We will show that all $\vee$ gates at level $h - 2$ have the same property (note that there are no $\vee$ gates in level $h - 1$). Let β_1 be an $\vee$ gate at level $h - 2$ labeled $[A, m_1, m_2]$. By construction, there is an $\wedge$ gate β_2 (β_1's parent) at level $h - 1$ labeled $[B, C, n_1, n_2, n_3]$, where $A = B$ or $A = C$ and $n_1 \leq n_3 < n_2$. Moreover, $n_1 \leq m_1 \leq m_2 \leq n_3$ if $A = B$ and $n_3 + 1 \leq m_1 \leq m_2 \leq n_2$ if $A = C$. Also by construction, there is an $\vee$ gate β_3 (β_2's parent) at level h labeled $[D, n_1, n_2]$. By inductive hypothesis, $i \leq n_1 \leq n_2 \leq k$. Therefore, $i \leq m_1 \leq m_2 \leq k$. $\square$

Corollary 2.1 All inputs to the circuit rooted at β are from the set $\{x_i, x_{i+1}, \ldots, x_k, \bar{x}_i, \bar{x}_{i+1}, \ldots, \bar{x}_k\}$.

Proof: By the above lemma, the interval indicated in the label of any $\vee$ gate at level 1 is of the form $[m, m], i \leq m \leq k$. The result follows from the input assignment rules stated above. $\square$

By a similar argument, the inputs to the circuit rooted at γ must be from the set $\{x_{k+1}, x_{k+2}, \ldots, x_j, \bar{x}_{k+1}, \bar{x}_{k+2}, \ldots, \bar{x}_j\}$.

The above results imply that for any $\wedge$ gate α in G_n with predecessors β and γ, the circuits rooted at β and γ have disjoint input sets.

Theorem 2.1 G_n is multilinear.

Proof: Suppose G_n is not multilinear. Then there must be an $\wedge$ gate α and a gate (or input) v in G_n such that there are multiple paths from v to α. Let β and γ be the two predecessors of α, then v must have paths to both of them. This implies that the input sets of the circuits rooted at β and γ must have a non-trivial intersection. But that is impossible by the above. □

2.2 The Canonical Formal Polynomial

Given a monotone Boolean circuit B_n computing a monotone function f, we establish some relationships between the monomials of $P(B_n)$ and the terms of PI(f) leading to a canonical form for $P(B_n)$. The proofs of the following lemmas are based on the idea that $P(B_n)$ and f must agree on every input assignment, since B_n computes f. In what follows, a *zero* monomial is one that has the constant 0, i.e., is identically zero.

Lemma 2.3 For each non-zero monomial ρ of $P(B_n)$, there exists a term $t \in$ PI(f) such that $var(t) \subseteq var(\rho)$.

Proof: Suppose there is a monomial ρ for which this claim is not true. On the input assignment that sets the variables in $var(\rho)$ to 1 and all the rest to 0, B_n evaluates to 1 but f is 0, leading to a contradiction. □

In the other direction, we have the following lemma.

Lemma 2.4 For all terms $t \in \mathrm{PI}(f)$, there exists a non-zero monomial ρ of $P(B_n)$, such that $var(t) = var(\rho)$.

Proof: Let t be any prime implicant of f. Consider the input that assigns 1 to the variables in t and 0 to all the rest. On this input, f evaluates to 1. Since B_n computes f, $P(B_n)$ must have a monomial ρ such that $var(\rho) \subseteq var(t)$, because otherwise B_n would evaluate to 0 on this input. There cannot be any other $t' \in \mathrm{PI}(f)$ such that $var(t') \subseteq var(\rho)$, for otherwise $var(t') \subseteq var(t)$ which is impossible since $t, t' \in \mathrm{PI}(f)$. It follows from lemma 2.3 that $var(\rho) = var(t)$. □

Since each parse-tree in B_n computes a monomial in $P(B_n)$, by the above lemmas there is a term in $\mathrm{PI}(f)$ associated with each parse-tree of B_n. By ordering the terms of $\mathrm{PI}(f)$, we associate a unique prime implicant with each parse-tree of B_n. This allows us to partition the set of parse-trees of B_n into *parse-classes*, $PC_1, \ldots, PC_s$, where $s = |\mathrm{PI}(f)|$. By lemma 2.4, each parse-class has at least one parse-tree whose variables correspond exactly with those of the prime implicant associated with the parse-class. We shall refer to one such parse-tree as a *representative* of the parse-class.

Thus, for any monotone B_n computing a monotone function f, we can put $P(B_n)$ in the following normal form: the monomials of $P(B_n)$ can be partitioned into $|\mathrm{PI}(f)|$ parse-classes; in each parse-class there are one or more monomials whose variable set coincides with that of the prime-implicant corresponding to the class and each of the rest of the monomials in the parse-class contains this set as a subset of its variable set. If B_n is restricted further to be multilinear, then none of the monomials have repeated literals and hence the terms corresponding to the representatives of each parse-class look exactly like the prime implicant corresponding to the class.

2.3 The Connectivity Function

Consider the function UCONN: $\{0,1\}^{n^2} \rightarrow \{0,1\}$, which takes as input the adjacency matrix of a graph and outputs a 1 if and only if the graph is connected. Now a graph is connected if and only if it has a spanning tree. Thus, the prime implicants of UCONN are the conjuncts corresponding to the spanning trees of a complete graph. Let $T_n = \{\, t : \{2,3,\ldots,n\} \rightarrow \{1,2,\ldots,n\} \mid \forall_i \exists_k t^k(i) = 1 \,\}$ and for each $t \in T_n$, let $p_t = \wedge_{i=2}^{n} x_{i,t(i)}$. Then PI(UCONN) is exactly $\{p_t \mid t \in T_n\}$. Each p_t is a tree rooted on the vertex labeled 1, spanning the complete graph on n vertices. It is well known that $|T_n| = n^{n-2}$.

UCONN can be computed by taking the transitive closure of the input adjacency matrix and performing $\wedge$ on the n^2 outputs. The circuit based on the well known algorithm by Floyd and Warshall (see [1]) for transitive closure is monotone and has size $O(n^3)$. Therefore, there is a $O(n^3)$ size monotone circuit for UCONN. In the next section we show that there are no polynomial size monotone multilinear circuits for UCONN. In particular, the above circuit based on the Floyd-Warshall algorithm is not multilinear.

3 A Lower Bound for Connectivity

In [7], Jerrum and Snir proved an exponential lower bound on the size of any Boolean circuit computing UCONN whose formal polynomial is of the form: $P = \sum_{t \in T_n} p_t$. We begin by showing that this lower bound can be extended to hold for any circuit whose formal polynomial is such that each non-zero monomial is exactly p_t, for some t. Such a polynomial differs from P in that some of the p_t's could occur more than once.

Definition 3.1 A Boolean circuit B_n is said to be *homogeneous* if all the non-zero monomials of $P(B_n)$ have the same number of positive variables. A monotone Boolean function is said to be

homogeneous with p-variables if each of its prime implicants has p variables.

Note that monotone functions computed by homogeneous circuits must be homogeneous. Note also that UCONN is a homogeneous function with $n-1$ variables.

Let $m = n^2$ and let B_m be a monotone, multilinear, homogeneous circuit for UCONN. Recall the canonical formal polynomial of a general Boolean circuit computing a monotone function (section 2.2). The monotonicity restriction removes the negative literals and multilinearity disallows any variable from occurring more than once in any monomial. By lemma 2.1, every non-zero monomial of $P(B_m)$ has at least $(n-1)$ variables. The homogeneity restriction allows only those monomials that have exactly $(n-1)$ variables, namely, those monomials whose variable sets correspond exactly to that of some prime implicant. Therefore,

$$P(B_m) = \sum_{t \in T_n} (p_t + p_t \ldots + p_t) + \text{zero monomials.}$$

In section 3.1 we show that B_m must have exponential size and in section 3.2 we show that any monotone multilinear circuit for UCONN can be converted into an equivalent monotone, multilinear, homogeneous circuit by only squaring the size. The lower bound follows.

3.1 Adaptation of Jerrum and Snir's Framework

As mentioned above, the main difference between B_m and the circuits considered in [7] is that B_m could have more than one parse-tree in B_m computing the same prime implicant p_t. To extend the lower bound, we simply fix n^{n-2} representative parse-trees $\{G_t \mid t \in T_n\}$ of B_m such that G_t computes p_t, for all t. The lower bound in [7] then applies to the sub-circuit of B_m that computes the disjunction of the monomials corresponding to the representative parse-trees. We have simplified the presentation in [7] to adapt it to the model we are considering.

Definition 3.2 For an $\wedge$ gate α, let $m(\alpha)$ be the number of representative parse-trees of B_m in which α appears.

Definition 3.3 An $\wedge$ gate α is said to be *(r,d)-significant* for $1 \leq r \leq n$ and $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$, if α appears in a parse-tree computing a term that has r variables, d of which are contributed by one of the immediate predecessors of α and $(r-d)$ by the other.

Definition 3.4 Let H be a subtree of a parse-tree T. Define weight of H as follows: $W(H) = \sum_{\alpha \in \wedge\text{-gates}(H)} \frac{1}{m(\alpha)}$, where $\wedge\text{-}gates(H)$ denotes the set of $\wedge$ gates in H.

Lemma 3.1 below corresponds to theorem 3.3 in [7].

Lemma 3.1 $\sum_{i=1}^{n^{n-2}} W(G_i) = |\{\alpha \in \wedge\text{-gates}(G_i) \mid m(\alpha) \geq 1\}|$.

Proof: By definition,

$$\sum_{i=1}^{n^{n-2}} W(G_i) = \sum_{i=1}^{n^{n-2}} \sum_{\alpha \in \wedge\text{-gates}(G_i)} \frac{1}{m(\alpha)}.$$

Fix an $\wedge$ gate α. For each representative parse-tree G_i, the contribution by α to the sum on the right-hand side of the above equation is either 0 (if α does not occur in it) or $\frac{1}{m(\alpha)}$. Thus, the total contribution by α is $m(\alpha)\frac{1}{m(\alpha)} = 1$ and therefore the right-hand side is the number of $\wedge$ gates α for which $m(\alpha) \geq 1$.□

The following lemma summarizes the arguments in section 4.5 of [7]. These arguments are presented in the appendix for the sake of completeness.

Lemma 3.2 [7] Let $n/2 < r \leq n-1$ and $1 \leq d < n/2$. If α is an (r,d)-significant $\wedge$ gate of B_m, then $m(\alpha) \leq (\frac{3n}{4})^{n-1}$.

Lemma 3.3 For any representative parse-tree G_t, $W(G_t) \geq (\frac{3n}{4})^{1-n}$.

Proof: Since G_t covers $(n-1)$ variables, there must be at least one (r,d)-significant $\wedge$ gate α in G_t such that $n/2 < r \leq (n-1)$ and $1 \leq d \leq (r-d) < n/2$. The proof then follows using lemma 3.2 since $W(H) \geq \frac{1}{m(\alpha)}$. □

From lemmas 3.1 and 3.3 it follows that any monotone, multilinear, homogeneous circuit for UCONN has size at least $n^{n-2}.(\frac{3n}{4})^{1-n}$. Therefore we have,

Theorem 3.1 Any monotone, multilinear, homogeneous Boolean circuit requires size $\geq \frac{1}{n} \cdot (\frac{4}{3})^{n-1}$ to compute UCONN.

3.2 Homogenizing Monotone Multilinear Circuits

Let B_n be a monotone multilinear Boolean circuit for UCONN. We present the construction of an equivalent monotone, multilinear, homogeneous circuit B'_n from B_n with size at most the square of the size of B_n. (Note that the main difference between B_n and B'_n is that there could be monomials in $P(B_n)$ that have more than $n-1$ variables.) An exponential lower bound on the size of monotone multilinear circuits for UCONN then follows from theorem 3.1.

Recall the canonical formal polynomial of a monotone multilinear Boolean circuit for a monotone function (section 2.1). Each parse-class has one or more monomials that look exactly like the prime implicant corresponding to the class. The construction below produces a circuit in which only these monomials survive. Thus, the resulting circuit is monotone, multilinear, homogeneous and computes the same function as the original circuit.

Lemma 3.4 Given a monotone multilinear circuit B_n of size s computing a homogeneous function f with p variables, there is a monotone, multilinear, homogeneous circuit B'_n that computes f within size $O(s^2)$.

Proof: Given B_n, we first construct an equivalent circuit C_n that has the following normal form: (i) C_n has alternating $\vee$ and $\wedge$ layers; (ii) the output is an $\vee$ gate and all circuit inputs are inputs to $\vee$ gates. It is easily verified that the size of C_n is at most twice that of B_n.

Given C_n, we construct an equivalent circuit B'_n such that each monomial of $P(B'_n)$ has exactly p variables. This is achieved by essentially keeping a count of the number of variables covered at a node.

- For every input A in C_n create an input A in B'_n.
- For every $\vee$ gate A in C_n create the $\vee$ gates $[A, i, 0]$, $0 \leq i \leq p$, in B'_n.
- For every $\wedge$ gate A in C_n create the $\vee$ gates $[A, i, 1]$, $0 \leq i \leq p$, in B'_n.
- For all $0 \leq i \leq p$, the inputs to an $\vee$ gate of the form $[A, i, 1]$ are $\wedge$ gates $[A, i, j, k]$, for all j, k such that $0 \leq j, k \leq p$ and $j + k = i$.
- For all i, j, k, inputs to the $\wedge$ gate $[A, i, j, k]$ are the $\vee$ gates $[B, j, 0]$ and $[C, k, 0]$, where B and C are the inputs of the $\wedge$ gate A in C_n.
- For all $0 \leq i \leq p$, the inputs to an $\vee$ gate of the form $[A, i, 0]$ are set as follows: for each input B of the $\vee$ gate A in C_n, (a) if B is an $\wedge$ gate, make $[B, i, 1]$ an input of $[A, i, 0]$, for all $0 \leq i \leq p$; (b) if B is an input labeled with x, $[A, 1, 0]$ has x as its input, and for all $i \neq 1$, $[A, i, 0]$ gets the constant 0 as an input; and (c) if B is an input labeled with a constant $c \in \{0, 1\}$, $[A, 0, 0]$ has c as its input, and for all $1 \leq i \leq p$, $[A, i, 0]$ gets the constant 0 as an input.

The size of B'_n is at most the square of that of C_n. It is also easily verified that the non-zero formal monomials of $P(B'_n)$ are those of $P(B_n)$ that have exactly p variables. Since the construction

preserves monotonicity and multilinearity, B'_n is a monotone, multilinear, homogeneous circuit computing f. □

From lemma 3.4 and theorem 3.1 we obtain the following theorem:

Theorem 3.2 Monotone multilinear circuits require size $\Omega(\sqrt{\frac{1}{n} \cdot (\frac{4}{3})^{n-1}})$ to compute UCONN.

3.3 Multilinearity vs. Polynomial Degree

We observe that there is a polynomial size monotone circuit for UCONN that also has polynomial degree. The lower bound in theorem 3.2 then establishes that polynomial degree circuits can be exponentially more powerful than multilinear circuits.

A Boolean circuit is said to be *semi-unbounded* [16] if each $\wedge$ gate in the circuit has in-degree 2 but $\vee$ gates could have in-degree more than 2. Polynomial size semi-unbounded circuits are also known to have polynomial degree [16]. The class $\mathcal{SAC}^1$ is characterized by uniform families of polynomial size, log-depth, semi-unbounded circuits. (The uniformity condition used is the notion of U_D-uniformity defined by Ruzzo [12].) The monotone analogue $m\mathcal{SAC}^1$ of $\mathcal{SAC}^1$ is then characterized by uniform families of monotone polynomial size, log-depth, semi-unbounded circuits [5].

Consider the reachability function USTCONN: $\{0,1\}^* \rightarrow \{0,1\}$, which takes as input the adjacency matrix of a graph G and two distinguished vertices s and t and outputs a 1 if and only if there is a path from s to t in G. USTCONN is known to be in $m\mathcal{SAC}^1$ (see [5]) and $\text{UCONN}(G) = \bigwedge_{1 \leq i,j \leq n} \text{USTCONN}(G,i,j)$. It is therefore easy to construct a $m\mathcal{SAC}^1$ circuit for UCONN by replacing the $\wedge$ gate of in-degree n^2 with an equivalent Boolean circuit of log-depth. Now $m\mathcal{SAC}^1$ circuits are known to have polynomial degree [16]. This implies that monotone polynomial degree circuits can compute UCONN within polynomial size, whereas monotone multilinear circuits require exponential size.

4 Concluding Remarks

In section 2.3 we talked about the $O(n^3)$ size monotone circuit based on Floyd-Warshall's algorithm for transitive closure. Each output of this circuit essentially computes the USTCONN function. A natural question is whether this circuit is multilinear. To answer this we observe that on an $n \times n$ adjacency matrix of a graph G as input, the $(1, n)$-th output of the Floyd-Warshall circuit tests all paths from 1 to n in G of length $\leq 2^n$. Note that the variables model the edges of the graph. Therefore monomials corresponding to the paths that are traverse the same edge more than once are not multilinear. The prime implicants of USTCONN are correspond to the edge-simple paths from 1 to n in G. These terms are multilinear and by lemma 2.2 each of these paths must be tested by the circuit.

Let us define a Boolean circuit to be *barely-multilinear* if there is at least one representative monomial in each parse-class that is multilinear. The Floyd-Warshall circuit for USTCONN is then barely-multilinear. That is, USTCONN is computable by polynomial size monotone barely-multilinear circuits. Now consider the construction in the proof of lemma 3.4. Since this construction essentially extracts multilinear monomials of length p, it also applies to monotone barely-multilinear circuits for homogeneous functions with p variables. That is, barely-multilinear circuits for UCONN can also be efficiently homogenized. This implies that the lower bound for UCONN in theorem 3.2 holds for barely-multilinear circuits as well. Thus, for monotone barely-multilinear circuits, UCONN is provably harder than USTCONN.

This contrasts Wigderson's observation in his survey on connectivity [18] that under almost any choice of reducibility, USTCONN is harder than UCONN. For example, in the Boolean decision tree model and under monotone p-projections, USTCONN is provably harder than UCONN [2, 14]. The only known exception is in the context of expressibility, where USTCONN is known to be a monadic

Σ_1^1 property[1] but UCONN is not [4].

Note that we do not know if there is a polynomial size monotone multilinear circuit for USTCONN. This is because, since USTCONN is not a homogeneous function lemma 3.4 is not applicable. We conclude with some of the questions that this work raises:

- Are there natural non-monotone functions in $\mathcal{P}$ for which multilinear circuits require exponential size?
- Is UCONN computable by polynomial size multilinear circuits that are not necessarily monotone? Is USTCONN computable by polynomial size monotone multilinear circuits?
- What is an appropriate notion of uniformity for multilinear circuits? Defining $\S\mathcal{P}$ to be the class of functions computable by uniform families of multilinear circuits within polynomial size, how does $\S\mathcal{P}$ compare with $\mathcal{NC}^1$? Is $\S\mathcal{P}$ closed under complement?

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading, Mass., 1974.

[2] M. Chrobak, H. Karloff and T. Radzik, *Connectivity vs. Reachability*, Information and Computation 91 (1991), 177-188.

[3] S. A. Cook, *A taxonomy of problems*, Information and Control 64 (1985), 2-22.

[4] R. Fagin, *Monadic generalized spectra*, Seitschrift fur Mathematische Logik und Grunlagen der Mathematik, 21 (1975), 89-96.

[1] Expressible as $\exists A_1 \exists A_2 \ldots \exists A_k \Psi$, where Ψ is a first order sentence and each A_i is a subset of the universe.

[5] M. Grigni and M. Sipser, *Monotone separation of Logspace from NC^1*, Proc. 6th IEEE Structures (1991), 294-298.

[6] J. E. Hopcroft and J. D. Ullman, *Introduction to automata theory, languages and computation*, Addison Wesley, Reading MA, 1979.

[7] M. Jerrum and M. Snir, *Some exact complexity results for straight-line computations over semi-rings*, J. Assoc. Comput. Mach. 29 (1982), 874-897.

[8] M. Karchmer and A. Wigderson, *Monotone circuits for connectivity require super-logarithmic depth*, SIAM J. Disc. Math. 3:2 (1990), 255-265.

[9] R. Raz and A. Wigderson, *Monotone circuits for matching require linear depth*, Proc. 22nd annual ACM STOC (1990), 287-292.

[10] A.A. Razborov, *A lower bound on the monotone network complexity of the logical permanent*, Mathematischi Zametki 37 (1985), 887-900.

[11] W. L. Ruzzo, *Tree-size bounded alternation*, J. Comput. Sys. Sci. 21:2 (1980), 218-235.

[12] W. L. Ruzzo, *On uniform circuit complexity*, J. Comput. Sys. Sci. 22:3 (1981), 365-383.

[13] J. E. Savage, *The complexity of computing*, R. E. Kreiger Publishing Co., Malabar, Florida, 1987.

[14] S. Skyum and L. Valiant, *A complexity theory based on Boolean algebra*, J. Assoc. Comput. Mach. 32 (1985), 484-502.

[15] É. Tardos, *The gap between monotone and non-monotone circuit complexity is exponential*, Combinatorica 7 (1987), 141-142.

[16] H. Venkateswaran, *Properties that characterize LOGCFL*, J. Comput. Sys. Sci. 43:2 (1991), 380-404.

[17] H. Venkateswaran, *Two dynamic programming algorithms for which interpreted pebbling helps*, Information and Computation 92:2 (1991), 237-252.

[18] A. Wigderson, *The complexity of graph connectivity*, Mathematical Foundations of Computer Science, LNCS 629 (1992), 112-132.

Appendix

We present the arguments in [7] that lead to lemma 3.2. Let B_m be a monotone multilinear circuit for UCONN.

Lemma 1 If α is an (r,d)-significant $\wedge$ gate of B_m, then $m(\alpha) \leq [\frac{(n-1)^2-d(r-d)-r(n-r-1)}{n-1}]^{n-1}$.

Proof: Let β and γ be the immediate predecessors of α in B_m. Let G be a representative parse-tree in which α appears with an r-variable term. Let a be the term formed at β and b be the term formed at γ such that $a \cdot b$ has $r \geq 1$ variables and a has $0 \leq d \leq \lfloor r/2 \rfloor$ variables that are not in b. Let c be a term such that $a \cdot b \cdot c$ is the $n-1$-variable term formed at the root of G. Clearly, c has $n-1-r$ variables that are not in a or b. Moreover, for every representative parse-tree that α participates in, the term formed at β (γ) must have exactly d ($r-d$, resp.) variables with the same set of row indices.

For $2 \leq i \leq n$, let X_i be the set of variables $\{x_{ij}\}$ such that x_{ij} appears in the monomial computed by some representative parse-tree that α participates in. Since the monomial computed by each representative parse-tree is exactly a prime implicant, $m(\alpha)$ is bounded above by the number of functions $t \in T_n$ such that $x_{i,t(i)} \in X_i$, for all $2 \leq i \leq n$. This in turn is at most the number of functions $t : \{2,3,\ldots,n\} \rightarrow \{1,2,\ldots,n\}$ such that $x_{i,t(i)} \in X_i$, for all $2 \leq i \leq n$. This number is exactly $\prod_{i=2}^{n} |X_i|$, since each term $\prod_{i=2}^{n} x_{ij}$, $x_{ij} \in X_i$, corresponds to a different function t. This product is maximized when $|X_i|$ is independent of i, therefore,

$$\prod_{i=2}^{n} |X_i| \leq [\frac{\sum_{i=2}^{n} |X_i|}{n-1}]^{n-1}.$$

The expression $\sum_{i=2}^{n} |X_i|$ is easily seen to be bounded above by $(n-1)^2$. This can be further refined to $(n-1)^2 - d(r-d) - r(n-r-1)$ by observing that the representative parse-trees of B_m cannot compute a monomial that contains both x_{ij} and x_{ji}, for any i,j. □

Let $c(r,d) = [\frac{(n-1)^2 - d(r-d) - r(n-r-1)}{n-1}]^{n-1}$. In the range $n/2 < r \leq (n-1)$ and $1 \leq d \leq (r-d) < n/2$, holding r constant, the expression is maximum at $d = r - \lceil n/2 \rceil + 1$ and allowing r to vary, the overall maximum is at $r = (n-1)$.

Therefore,

$$\begin{aligned} c(r,d) &\leq [\frac{(n-1)^2 - n/2(n/2-1)}{n-1}]^{n-1} \\ &= [\frac{3n^2/4 - 3n/2 + 1)}{n-1}]^{n-1} \\ &\leq (\frac{3n}{4})^{n-1} \end{aligned}$$

This in conjunction with lemma 1 above leads to lemma 3.2.

Cancellation Is Exponentially Powerful for Computing the Determinant

Rimli Sengupta*
Dept. of Computer Science
Rose-Hulman Institute of Technology
Terre Haute, IN 47803
e-mail : rimli@cs.rose-hulman.edu

March 26, 1997

Abstract

By defining a notion of cancellation in algebraic circuits, we study the power of negation at a finer granularity than previously considered. Our result is that in the absence of cancellation, computing the determinant requires exponential circuit size. Previous work shows that cancellations are essential for efficiently computing certain monotone (all coefficients positive) polynomials. The work presented in this paper extends that to the computation of non-monotone polynomials and shows that it is the cancellative aspect of negation that allows efficient computation of even certain non-monotone polynomials. We present the first example of a non-monotone polynomial for which cancellations lead to exponential savings in circuit size: determinant.

Keywords: Algebraic circuit, cancellation, determinant.

1 Introduction

Using the power of cancellation to compute efficiently has been a recurrent theme in the history of algebraic algorithm design. An early example of this is integer multiplication. Two n-digit integers can be multiplied trivially in $O(n^2)$ steps. Making clever usage of cancellation, Karatsuba and Ofman [4] obtained a $O(n^{1.59})$ divide-and-conquer algorithm. In a breakthrough result, Strassen [7] made elegant usage of cancellation to obtain a surprising $O(n^{2.81})$ algorithm for matrix multiplication, improving the obvious $O(n^3)$ algorithm. Schnorr later showed that the $O(n^3)$ algorithm is the best possible in the absence of negation [6]. This gap between Schnorr's lower bound and Strassen's upper bound was, to our knowledge, the first formal evidence of the power of cancellation in algebraic computation. In [8], Valiant widened this gap to exponential by showing that algebraic circuits can count the number of perfect matchings in triangular grid graphs within polynomial size in the presence of negation, but require exponential size in the absence thereof.

All of these results are about using cancellations to efficiently compute polynomials with positive coefficients (monotone polynomials). For computing monotone polynomials, studying the power of cancellation is equivalent to studying the power of negation since negations can only be used for cancellation. The situation is quite different in the case of polynomials that may have negative coefficients (non-monotone polynomials). There is an important distinction between negation and

*Part of this work was done while the author was at the College of Computing, Georgia Tech, Atlanta GA 30332-0280, which was supported by NSF grant CCR-9200878.

cancellation for computing non-monotone polynomials: negations are essential but cancellations are not. Since negations and cancellations have historically been viewed as a unit, the question as to whether cancellations help in computation of non-monotone polynomials has not been studied before. To investigate the power of cancellation for computing non-monotone polynomials we identify a notion of cancellation based on additive inverse. Our result is that in the absence of cancellation, computing the determinant using circuits over reals requires exponential size. Since the determinant function is known to be computable within size $O(n^{3.5})$ using circuits over reals [1], our lower bound for the determinant implies that cancellations can be exponentially powerful even for computing non-monotone polynomials. This result also provides an important insight about efficient computation of the determinant [1]: negations (-1) have to be supplied since they are essential for computing the determinant, but this results in the automatic availability of cancellation which causes the computation to be efficient.

Cancellations appear in the study of the power of commutativity in algebraic circuits [2, 5]. Nisan [5] studied non-commutative circuits over reals for computing the determinant. Non-commutative circuits cannot perform cancellation of terms with different multiplicative order ($(xy-yx)$-type cancellations) but can perform cancellation of terms with same multiplicative order ($(xy-xy)$-type cancellations). Nisan showed that non-commutative circuits for computing the determinant require exponential formula size and observed that efficient commutative formulas for the determinant must rely on $(xy-yx)$-type cancellations. Withholding commutativity essentially amounts to withholding $(xy-yx)$-type cancellations. We note that Strassen's method for multiplying two 2×2 matrices only uses $(xy-xy)$-type cancellations and is therefore recursively applicable to matrices, leading to the $O(n^{2.81})$ matrix multiplication algorithm. This illustrates that $(xy-xy)$-type cancellations alone can be quite useful. Nisan's work on non-commutative circuits over reals therefore does not settle the question about the power of cancellations in computing the determinant using circuits over reals. We consider circuits over reals that do not use either type of cancellation, but do not have to be non-commutative and show that computing the determinant in this model requires exponential circuit size. While this is stronger than the exponential formula size bound obtained by Nisan in the non-commutative model, the question as to whether computing the determinant requires exponential circuit size in the non-commutative setting remains open.

2 Arithmetic Circuits

An arithmetic circuit A_n over the field $\mathcal{R}$ of reals is a directed acyclic labeled graph in which nodes either have in-degree 0 or 2. Nodes with in-degree 0 are called *inputs* and are labeled from the set $\{x_i|\ 1\le i\le n\}\cup\{-1\}$. The other nodes (also called *gates*) are labeled from the set $\{+,\times\}$. The circuit has exactly one node with 0 out-degree and it is called the *output*. A *formula* is a circuit in which no gate has out-degree >1. If no input is labeled with the constant -1, the arithmetic circuit is said to be *monotone*. The *size* of an arithmetic circuit is the number of gates in it and its *depth* is the length of the longest path from any input to the output.

Each finite A_n computes a polynomial function $f:\mathcal{R}^n\to\mathcal{R}$. A *term* is a product of elements from the set $\{x_i|\ 1\le i\le n\}\cup\{-1\}$. We shall use $var(t)$ and $sign(t)$ to denote the set of variables in a term t and its sign, respectively.

Definition 2.1 A *parse-graph* G of a circuit A_n is defined inductively as follows: G includes the output of A_n; for any $+$ gate v included in G, exactly one immediate predecessor of v in A_n is included as its only predecessor in G; and for any $\times$ gate v included in G, both the immediate predecessors of v in B_n are included as its predecessors in G.

Every gate v of a parse-graph G computes a term which is the product of the labels on the inputs of the sub-graph rooted at v. With each A_n we associate a formal polynomial $P(A_n)$, which is the sum of the terms computed at the root of each parse-graph of A_n.

Definition 2.2 Let $\mathcal{G}$ denote the set of parse-graphs of A_n and let $t(G)$ be the term computed at the output of parse-graph G. The *formal polynomial* $P(A_n)$ of a circuit A_n is

$$\sum_{G \in \mathcal{G}} t(G).$$

We note the monomials of $P(A_n)$ have coefficients equal to ± 1.

2.1 Cancellations in Arithmetic Circuits

The notion of cancellation in arithmetic circuits is intuitive. Monomials ρ, ρ' in $P(A_n)$ are said to *cancel* if $\rho + \rho'$ is identically zero.

Definition 2.3 An arithmetic circuit A_n is said to be *non-cancellative* if there are no monomials in $P(A_n)$ that cancel.

We note that non-cancellative arithmetic circuits can compute polynomials with negative coefficients. That is, while negation (in the form of the constant -1) is essential to computing polynomials with negative coefficients, cancellations are not. In the next section we consider non-cancellative arithmetic circuits that compute the determinant function.

3 A Lower Bound for the Determinant Function

Consider the determinant function DET: $\mathcal{R}^{n^2} \rightarrow \mathcal{R}$ defined as follows:

$$\text{DET}[X] = \sum_{\pi \in S_n} sign(\pi) \cdot \prod_{i=1,n} x_{i,\pi(i)},$$

where $X = [x_{ij}]$ is an $n \times n$ matrix, S_n is the set of all permutation functions on n elements, and $sign(\pi)$ is $+1(-1)$ if permutation π has an odd (even, resp.) number of cycles. Let p_π denote the term $\prod_{i=1,n} x_{i,\pi(i)}$, for $\pi \in S_n$.

3.1 The Formal Polynomial

Throughout this section we will assume that A_{n^2} is a non-cancellative arithmetic circuit computing DET. We first establish a canonical form for $P(A_{n^2})$. The proofs of the lemmas in this section are based on the idea that since A_{n^2} computes DET, $P(A_{n^2})$ and DET must agree on every input assignment.

A multivariate polynomial is said to be multilinear if none of the variables occur more than once in any monomial. Note that DET$[X]$ is a multilinear polynomial. We begin by showing that for any non-cancellative circuit A_{n^2} for computing DET, the polynomial $P(A_{n^2})$ must be multilinear.

Theorem 3.1 $P(A_{n^2})$ is multilinear.

Proof: Since A_{n^2} computes DET,

$$P(A_{n^2}) - \text{DET}[X] = 0,$$

for all possible input assignments to the n^2 variables of X. The equality can only hold if the multinomial on the left-hand side (l.h.s) is identically zero. Supposing $P(A_{n^2})$ is not multilinear let ρ be a term in $P(A_{n^2})$ in which a variable occurs k times, $k > 1$. Since all the terms in the l.h.s have coefficients equal to ± 1, there must be a term ρ' in the l.h.s such that ρ and ρ' cancel each other. But ρ' cannot belong to DET$[X]$ since it must have a variable that occurs $k > 1$ times and DET$[X]$ is multilinear. Therefore it must come from $P(A_{n^2})$, thereby violating the assumption that A_{n^2} is non-cancellative. □

We use this theorem to derive a criterion for cancellation that we use in the proof of the following lemmas.

Corollary 3.1 There are no two terms ρ and ρ' in $P(A_{n^2})$ with $var(\rho) = var(\rho')$ and $sign(\rho) \neq sign(\rho')$.

Proof: Since $P(A_{n^2})$ is multilinear, the existence of two such terms would mean that they would cancel each other, which is impossible since A_{n^2} is non-cancellative. □

We now determine the variable sets of the monomials in $P(A_{n^2})$.

Lemma 3.1 For each monomial ρ in $P(A_{n^2})$, $var(p_\pi) \subseteq var(\rho)$ for some $\pi \in S_n$.

Proof: Suppose there is a monomial ρ in $P(A_{n^2})$ such that there is no $\pi \in S_n$ with $var(p_\pi) \subseteq var(\rho)$. Let ρ' be a monomial in $P(A_{n^2})$ such that $var(\rho') \subseteq var(\rho)$ and $var(\rho')$ is an inclusion minimal set for which this containment holds. On the input assignment that sets the variables in $var(\rho')$ to 1 and the rest to 0, DET evaluates to 0 but $P(A_{n^2}) \neq 0$, since A_{n^2} is non-cancellative and therefore ρ' is not cancelled. □

Lemma 3.2 For all $\pi \in S_n$, there is a unique $\rho \in P(A_{n^2})$ such that $var(\rho) = var(p_\pi)$ and $sign(\rho) = sign(\pi)$.

Proof: To show existence, suppose there is a $\pi \in S_n$ such that there is no monomial ρ in $P(A_{n^2})$ with $var(\rho) = var(p_\pi)$ and $sign(\rho) = sign(\pi)$. Consider the input assignment that sets the variables in $var(p_\pi)$ to 1 and all the rest to 0. DET evaluates to ± 1 on this input. For $P(A_{n^2})$ to be non-zero on this input, there must exist a monomial ρ in $P(A_{n^2})$ such that $var(\rho) \subset var(p_\pi)$. By lemma 3.1, this implies the existence of a $\sigma \in S_n$ such that $var(p_\sigma) \subset var(p_\pi)$, which is impossible.

To show uniqueness, suppose there exist monomials ρ and ρ' in $P(A_{n^2})$ such that $var(\rho) = var(\rho') = var(p_\pi)$. Since A_{n^2} is non-cancellative, ρ and ρ' are not cancelled and therefore $sign(\rho) = sign(\rho')$. On the input assignment that sets the variables in $var(p_\pi)$ to 1 and all the rest to 0, DET evaluates to ± 1 but $P(A_{n^2})$ evaluates to $\pm k$, for $k \geq 2$.

Finally, if $sign(\rho) \neq sign(\pi)$, then $sign(P(A_{n^2})) \neq sign(\text{DET})$ on the above input. □

By lemma 3.1, each monomial of $P(A_{n^2})$ contains the variable set of p_π, for some permutation π. Conceivably, some of them could contain the variable sets corresponding to more than one permutation. By ordering the permutations, we can associate a unique permutation with each monomial. Thus, for any non-cancellative arithmetic circuit A_{n^2} computing DET, we can put $P(A_{n^2})$ in the following normal form by the above lemmas: the monomials of $P(A_{n^2})$ can be

partitioned into $n!$ classes $\{C_\pi \mid \pi \in S_n\}$; in each class C_π, there is a unique monomial ρ with $var(\rho) = var(p_\pi)$. Using lemma 3.2, lemma 3.1 can be improved to show that in fact there are no monomials in $P(A_{n^2})$ whose variable sets do not equal $var(p_\pi)$, for some $\pi \in S_n$. That is, each class C_π is a singleton. To prove the following lemma we shall use the following readily verifiable fact:

Fact 1: For any $\pi, \sigma \in S_n$, $|var(p_\pi) - var(p_\sigma)| \geq 2$.

Lemma 3.3 For every monomial ρ in $P(A_{n^2})$, $var(p_\pi) = var(\rho)$, for some $\pi \in S_n$.

Proof: By lemma 3.1, for each ρ in $P(A_{n^2})$, $var(\rho) = var(p_\pi) \cup V_\rho$, for some $\pi \in S_n$ and for some variable set V_ρ with $V_\rho \cap var(p_\pi) = \emptyset$. We prove the lemma by showing that $|V_\rho| \neq k, k \geq 1$, for any ρ in $P(A_{n^2})$. The proof is by induction on k.

(Base Case:) k=1. Let $var(\rho) = var(p_\pi) \cup \{x\}$. Consider the input assignment that sets the variables in $var(\rho)$ to 1 and the rest to 0. By fact 1, π is the only permutation such that $var(p_\pi) \subseteq var(\rho)$. Therefore, DET evaluates to $sign(\pi)$ on this input. By lemma 3.2, there is a unique monomial in $P(A_{n^2})$ with variable set $var(p_\pi)$, which also evaluates to $sign(\pi)$ on this input. Since π is the only permutation such that $var(p_\pi) \subseteq var(\rho)$, the only other monomials in $P(A_{n^2})$ that evaluate to non-zero on this input are those with variable set exactly $var(\rho)$. Since A_{n^2} is non-cancellative, these monomials must all have the same sign. It then follows that $P(A_{n^2}) \neq$ DET on this input, regardless of the sign of π. Therefore, there cannot be a monomial ρ in $P(A_{n^2})$ with $|V_\rho| = 1$.

(Inductive step:) By inductive hypothesis, for all monomials ρ, $|V_\rho| \neq r$, for $1 \leq r \leq (k-1)$. Let $var(\rho) = var(p_\pi) \cup V_\rho$, such that $|V_\rho| = k$. Consider the input that sets the variables in $var(\rho)$ to 1 and the rest to 0. Suppose $var(p_\pi) \cup V_\rho$ spans the variable sets of $q+s$ permutations, q of them with positive sign and the rest with negative sign. On this input, DET evaluates to $q-s$. By lemma 3.2, there are $q+s$ monomials in $P(A_{n^2})$ that collectively evaluate to $q-s$. All the other monomials of $P(A_{n^2})$ that are non-zero on this input must collectively evaluate to 0, for DET and $P(A_{n^2})$ to agree. Since ρ is non-zero on this input, there must be a monomial ρ', with $sign(\rho) \neq sign(\rho')$, that is non-zero on this input. For ρ' to be non-zero on this input, $var(\rho') \subseteq var(\rho)$. By lemma 3.1, let $var(\rho') = var(p_\sigma) \cup V_{\rho'}$, for some $\sigma \in S_n$, with $V'_\rho \cap var(p_\sigma) = \emptyset$. Since only $n+k$ variables were set to 1, $|V_{\rho'}| \leq k$. If $|V_{\rho'}| = k$, then $var(\rho')$ must equal $var(\rho)$. Since $sign(\rho) \neq sign(\rho')$, ρ and ρ' must cancel, which is impossible since A_{n^2} is non-cancellative. Thus $|V_{\rho'}| < k$, violating the inductive hypothesis. □

Based on the above lemmas, we can completely determine the variable set and sign of each monomial in $P(A_{n^2})$. Moreover, by theorem 3.1 $P(A_{n^2})$ must be multilinear as well and we have,

Theorem 3.2 For any non-cancellative arithmetic circuit A_{n^2} computing DET,

$$P(A_{n^2}) = \sum_{\pi \in S_n} sign(\pi) \cdot p_\pi.$$

3.2 Determinant versus Permanent

The permanent of a matrix $X = [x_{ij}], 1 \leq i, j \leq n$ can be defined as,

$$\text{PERM}[X] = \sum_{\pi \in S_n} p_\pi,$$

where S_n and p_π are as defined above. The following theorem is easily obtained using arguments very similar to the ones in the lemmas above.

Theorem 3.3 For any non-cancellative arithmetic circuit A_{n^2} computing PERM,

$$P(A_{n^2}) = \sum_{\pi \in S_n} p_\pi.$$

Thus, non-cancellative arithmetic circuits for PERM must also be monotone. This is consistent with the intuition that negations can only be used for cancellation while computing monotone polynomials. In [3], Jerrum and Snir had shown an exponential size lower bound for monotone arithmetic circuits that compute PERM.

Theorem 3.4 [3] Monotone arithmetic circuits for PERM require size $\geq n(2^{(n-1)} - 1)$.

The lower bounds now follow from theorems 3.2, 3.3, 3.4.

Theorem 3.5 Non-cancellative arithmetic circuits for PERM require size $\geq n(2^{(n-1)} - 1)$.

Theorem 3.6 Non-cancellative arithmetic circuits for DET require size $\geq n(2^{(n-1)} - 1)$.

Proof: Let A_{n^2} be a non-cancellative arithmetic circuit that computes DET within size $s < n(2^{(n-1)} - 1)$. The circuit of size s obtained by relabeling each -1 input of A_{n^2} with 1 is monotone and by theorem 3.2, computes PERM, thereby violating theorem 3.4. □

Since DET is known to be computable within size $O(n^{3.5})$ using arithmetic circuits [1], the above lower bound establishes that cancellations can be exponentially powerful, even for computing polynomials with negative coefficients. Note that PERM is not known to be computable by polynomial size arithmetic circuits.

In [5] Nisan had shown that in the non-commutative setting, computing the determinant is as hard as computing the permanent. Our results show that this is true in the non-cancellative setting as well. Nisan had proven exponential formula size lower bounds for these functions in the non-commutative setting. We are able to show exponential circuit size lower bounds for these functions in the non-cancellative setting. The question as to whether these functions require exponential circuit size in the non-commutative setting remains open.

Acknowledgements The author thanks H. Venkateswaran for the discussions that led to this work. She also thanks Steve Capell for several helpful suggestions.

References

[1] S. J. Berkowitz, *On computing the determinant in small parallel time using a small number of processors*, Information Processing Letters 18 (1984), 147-150.

[2] L. Hyafil, *The power of commutativity*, Proc. 18^{th} IEEE FOCS (1977), 171-174.

[3] M. Jerrum and M. Snir, *Some exact complexity results for straight-line computations over semi-rings*, J. Assoc. Comput. Mach. 29 (1982), 874-897.

[4] A. Karatsuba and Y. Ofman, *Multiplication of multi-digit numbers on automata*, Dokl. Akad. Nauk SSSR 145 (1962), 293-294.

[5] N. Nisan, *Lower bounds for non-commutative computation*, Proc. 23nd annual ACM STOC (1991), 410-418.

[6] C. P. Schnorr, *A lower bound on the number of additions in monotone computations*, Theor. Comput. Sci. 2 (1976), 305-315.

[7] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. 13 (1969) 354-356.

[8] L. Valiant, *Negation can be exponentially powerful*, Theoretical Computer Science 12 (1980), 303-314.